Медников Дмитрий Николаевич

ИНН 773703126905, адрес: 109004,РФ, г. Москва, Тетеринский переулок, дом 4, стр. 2, пом.30

Программа для ЭВМ
Программный комплекс ВМ

Инструкция пользователя

Москва 2023

Оглавление

1. Введение……4

2. Назначение системы……4

3. Назначение документа……4

4. Уровень подготовки пользователей……4

5. Назначение и условия применения Программы……..4

b. Программные и аппаратные требования к системе……4

c. Рекомендованные настройки безопасности веб- браузеров……5

6. Подготовка к работе, работа с Программой……..5

a. Вход в систему……..5

b. Авторизация в системе……6

7. Описание и настройка АРМ (автоматизированного рабочего места) и главного меню……6

7.1 Термины и определения……..6

7 7.2 Главное меню платформы……6

7.3 Определение состава АРМ……..8

7.4 Функциональные модули……..12

8. Настройка Представлений……...16

8.1 Общая информация……16

8.2 Общий интерфейс……17

8.3 Действия с представлениями……21

8.4 Форма редактирования представления……....23

8.5 Создание параметров……33

8.6 Создание кнопок……..37

8.7 Работа с данными представления……...38

8.8 Дочерние представления……39

8.9 Формирование запросов в представлении……40

8.10 Chart для представлений……41

9. Работа с рядами данных……42

9.1 Загрузка нового ряда……..43

9.2 Загрузка в уже существующий ряд……44

9.3 Доступные для использования библиотеки Python……45

9.4 Расчет статистики……46

9.5 Создание новой статистики……..47

9.6 Изменение колонок таблицы……48

9.7 Конфигурация action 9.8 Описание структуры конфига……49

9.10 Анализ рядов……...51

9.11 Генерация кода……...52

9.12 Соединение рядов……53

9.13 Расчет для показателя……..……..54

9.14 Доступ к показателям по имени……55

9.15 Доступ к базе данных……55

9.16 Доступ к данным документа, листа и анализа……55

9.17 Мониторинг результатов вычисления……..56

9.18 История изменений……57

9.19 Периодический пересчет данных ряда……58

10. Завершение работы с Программой 11. Возникновение вопросов или внештатных ситуаций……59

1. Введение

Документ содержит информацию, необходимую для эксплуатации программы для ЭВМ Программный комплекс ВМ (далее - Программа). В документе представлена последовательность действий для запуска, выполнения и завершения Программы.

2. Назначение системы

Программа для ЭВМ Программный комплекс ВМ предназначена для:

Разработки веб приложений для решения широкого спектра задач в сфере аналитики, обработки и отображения данных.

3. Назначение документа

Документ описывает порядок работы пользователя с системой.

4. Уровень подготовки пользователей

Пользователь Платформы должен иметь навык работы с любым из поддерживаемых интернет браузеров (Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Internet Explorer), а также знать соответствующую предметную область.

5. Назначение и условия применения Программы

Программа предназначена для решения следующих задач:

- Создания и настройки АРМ пользователя
- Создания и настройки представлений
- Создания и настройки форм обновления данных
- Получения данных из внешних источников
- Хранения информации
- Обработки и анализа данных
- Управления бизнес-процессами
- Формирования отчетности

b. Программные и аппаратные требования к системе

Для корректной работы с платформой необходима следующая конфигурация автоматизированного рабочего места пользователя.

Минимальные требования к системе:

- 4 ядра
- Из расчета 25 пользователей на 1 ядро для расширения
- 4 ГБ доступной памяти на 1 ядро системы

Поддерживаемые ОС:

- Microsoft Windows (32-bit or 64-bit)
- Apple Mac OS
- Fedora
- Debian Linux
- HP-UX
- FreeBSD
- CentOS
- Ubuntu

Поддерживаемые веб-браузеры.

- Mozilla Firefox
- Microsoft Internet Explorer
- Apple Safari
- Google Chrome

c. Рекомендованные настройки безопасности веб- браузеров:

- Cookies
- Pop-ups (new windows/tabs)
- Javascript
- AJAX
- DHTML

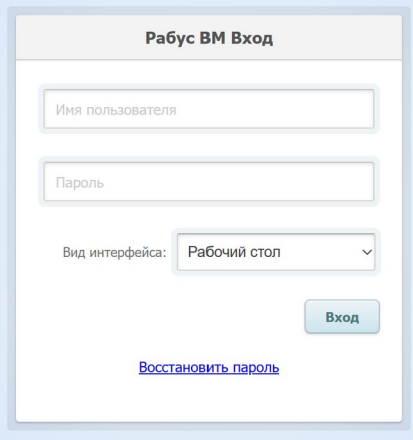
6. Подготовка к работе, работа с Программой

a. Вход в систему

Для входа в систему запустите браузер и наберите в адресной строке URL-адрес портала.

https://bm.t48.ru
Учетная запись для доступа:
логин: BM
пароль: !bm2023!

Форма авторизации в системе



b. Авторизация в системе

Для авторизации введите в поля Пользователь и Пароль, соответствующие данные и нажмите кнопку «Вход».

При попытке доступа к порталу с неверными данными возникает ошибка: «Неверно указан логин/пароль».

После удачной авторизации, будет осуществлен переход на Домашнюю страницу портала.

7. Описание и настройка АРМ (автоматизированного рабочего места) и главного меню

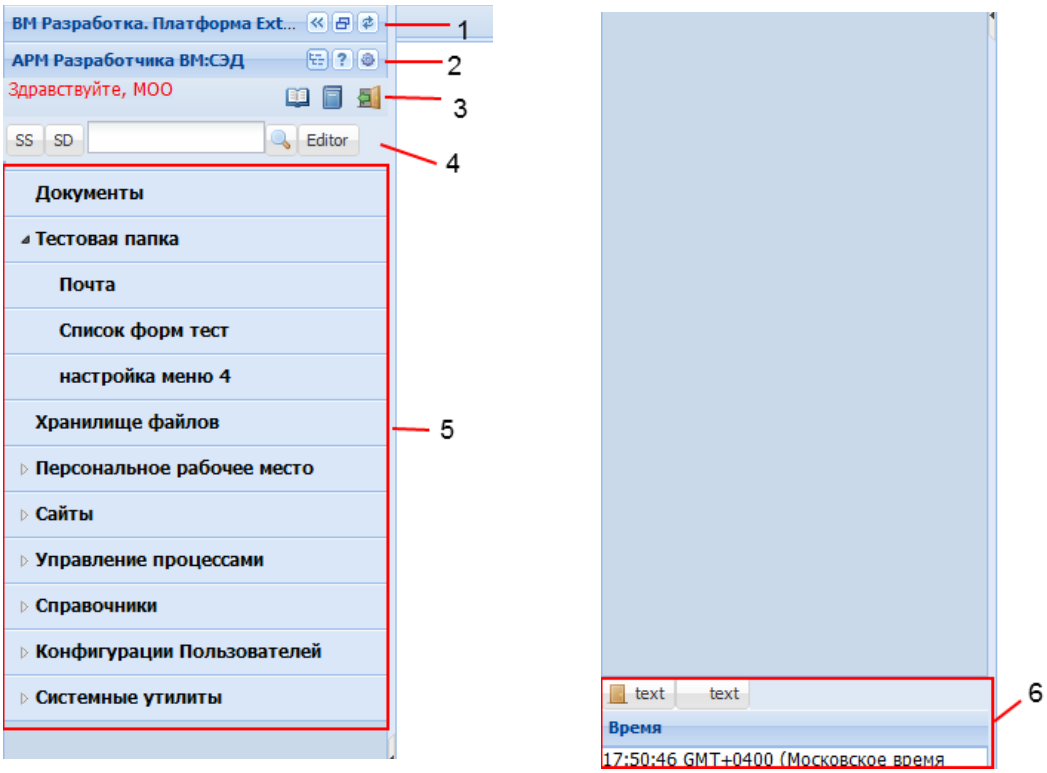
7.1 Термины и определения

Термин	Описание
Меню	Инструмент, позволяющий выбрать из соответствующего списка необходимую страницу или экран и открыть в рабочей области.
Папка	Элемент меню, содержащий различные АРМ, или другие папки.
АРМ	Содержит разделы и пункты меню, предназначенные для данного АРМ.
Раздел меню	Содержит разделы и пункты меню.
Пункт меню	Ссылка на конкретную страницу или экран.
Портлет	Находящиеся в панели меню функции, формы, представления, или кнопки, облегчающие работу пользователя конкретного АРМ.
Главное меню	Панель в левой части страницы, содержащая меню, портлеты, и

	панель инструментов.

7.2 Главное меню платформы

Вид интерфейса «Интеллектуальное меню»



Главное меню состоит из следующих областей:

№	Наименование	Что делает
1	панель «Решение»	Верхняя панель меню, содержащая название текущей версии БМ. Содержит панель инструментов, позволяющую *Свернуть* меню, *Закрыть все открытые окна*, а так же *обновить меню и другие элементы панели.*
2	Панель «Роль»	Показывает название текущего АРМ (Автоматизированного рабочего места), а так же предоставляет следующие возможности: «*Выбор роли*» - показать меню другой роли(должности). Доступно, если сотрудник назначен на несколько должностей; «*Справка*» - открыть окно документации; «*Настройка пользовательской конфигурации*» -

		открыть окно настройки. Настройка позволяет выбрать тему, размер шрифтов, и др.
3	Панель управления меню.	Содержит кнопки «Раскрыть» и «Свернуть», которые раскрывают и сворачивают все разделы меню, а так же кнопку «Выйти из системы»
4	Верхняя панель портлетов.	Содержит динамически подключаемые к данному АРМ портлеты, представляющие собой области отображения данных. Могут быть показаны данные, которые являются результатом выполнения функции, или набор инструментов (кнопки), облегчающие работу пользователя. Также в портлете возможно отобразить форму, представление или экран. Экраны могут содержать графики, рисунки и др. информацию
5	Главное Меню.	Содержит элементы меню данного АРМ. Это могут быть разделы и пункты меню. Раздел меню раскрывается по одиночному или двойному нажатию, и может содержать как другие разделы, так и пункты меню. Пункт меню, открывает вкладку или окно в рабочей области программы, в которой отображаются экраны, представления и др. Для выполнения работ Пользователь выбирает пункт меню.
6	Нижняя панель портлетов.	Аналогично верхней, содержит динамически подключаемые к данному АРМ портлеты, представляющие собой функции, формы, представления, или кнопки, облегчающие работу пользователя.

7.3 Определение состава АРМ

Состав АРМ.

Состав меню АРМ определяется по результатам работ по обследованию компании. Каждый пункт меню отображает необходимый набор данных и изменяется администратором, подстраивая меню под потребности компании. В любой момент администратор может изменять состав меню по запросу пользователя.

Начальное состояние.

Состав АРМ можно создать, используя инструмент «Настройка меню». Мы создаем запись, соответствующую АРМ. Находясь на этой записи добавляем «раздел меню» или «пункт меню».

Процесс создания нового объекта.

Добавление разделов меню и пунктов меню производится с помощью нажатия кнопки в окне «Меню»

Новые элементы меню создаются в окне «Настройка меню» путем нажатия на кнопку «Создать новый объект» (кнопка в левом верхнем углу).

По нажатию этой кнопки появляется окно, в котором можно задать такие параметры как название, тип элемента, порядок, в соответствии с которым элемент расположится в списке. а так же в случае, если создается АРМ, создать требуемые портлеты в любом количестве. Редактирование элементов осуществляется путем двойного клика по элементу в списке, либо, если выбрана вкладка «Параметры вызова», по одиночному нажатию.

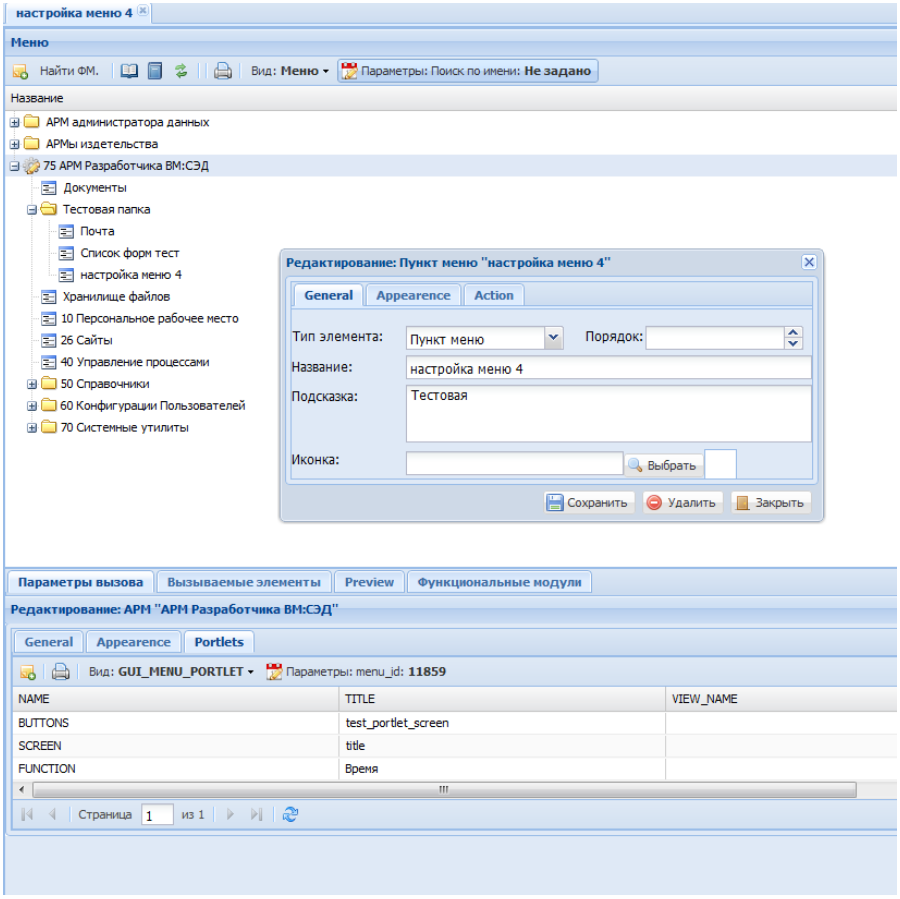


Рис. 1

Вкладка General.

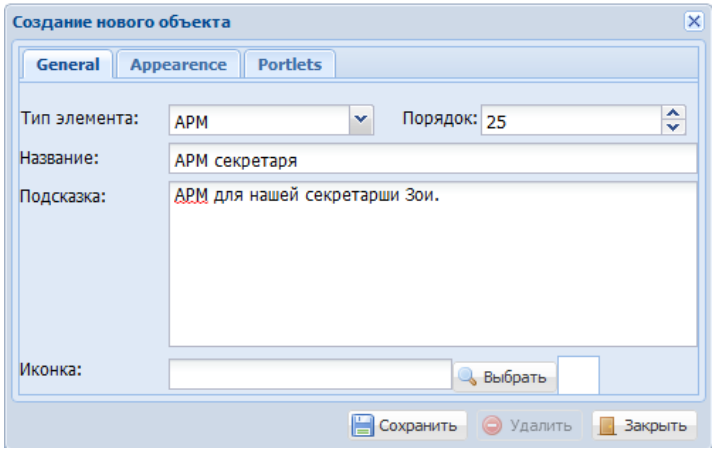


Рис. 2 Вкладка General.

Тип элемента. Позволяет выбрать один из 4 типов элементов: Папку, АРМ, раздел меню, или пункт меню. Папку можно создать только в корневом каталоге или в другой папке, равно как и АРМ. Внутри АРМа можно создать разделы и пункты меню. Внутри раздела меню создается только пункт меню или другой раздел.

Порядок определяет положение элемента в списке.

Если мы выберем пункт меню с уже существующим порядком и попытаемся создать новый элемент, то он будет создан рядом с выбранным пунктом, и его «Порядок» будет на единицу больше порядка соседнего элемента (в дереве элементов будет находиться на одну позицию ниже).

Название отображается в главное меню.

Иконка позволяет выбрать уже существующую иконку, либо загрузить новую.

Вкладка Appearence.

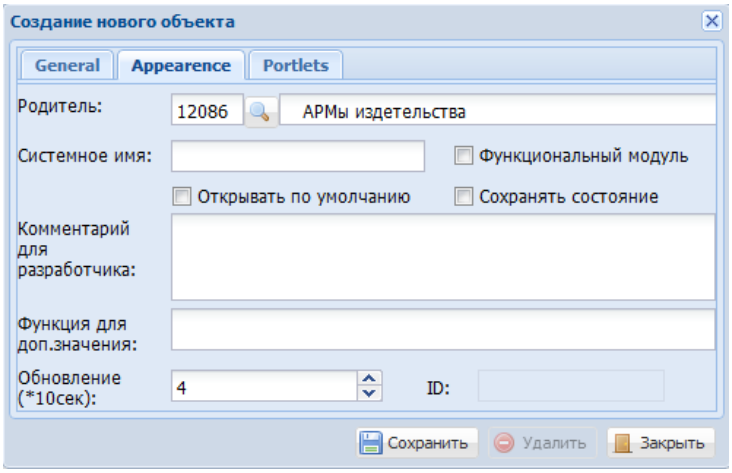


Рис. 3 Вкладка Appearence.

На этой вкладке мы видим служебные данные пункта меню.

Родитель: ID и имя родительского элемента.

Функциональный модуль: Флаг, определяющий «функциональный модуль»

Открывать по умолчанию. Если установлен флажок, то папка, АРМ, или раздел меню будут открыты сразу после загрузки главного меню, а пункт меню будет запущен в рабочей области.

Вкладка Action.

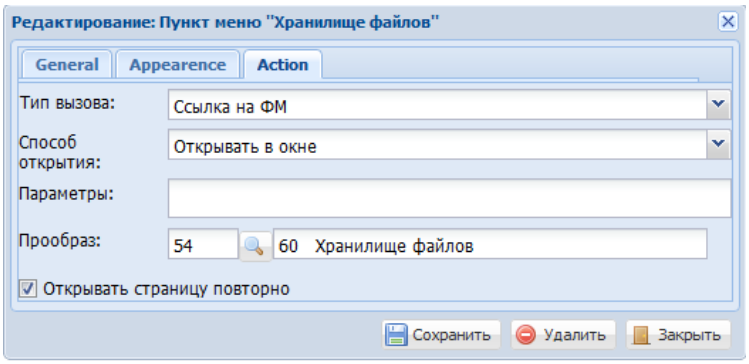


Рис. 4 Вкладка Action.

Вкладка Action активна только для пунктов и разделов меню.

Тип вызова: URL (ссылка на сайт), Экран, представление, JS-файл, ссылка на функциональный модуль.

Способ открытия: Позволяет задать открытие в окне, во вкладке, а так же в браузере(?).

Прообраз/URL/Представление: Данный пункт различен для каждого типа вызова. Он однозначно характеризует вызываемый объект, т.е. Позволяет нам указать конкретное представление или URL необходимого объекта.

Вкладка Portlets.

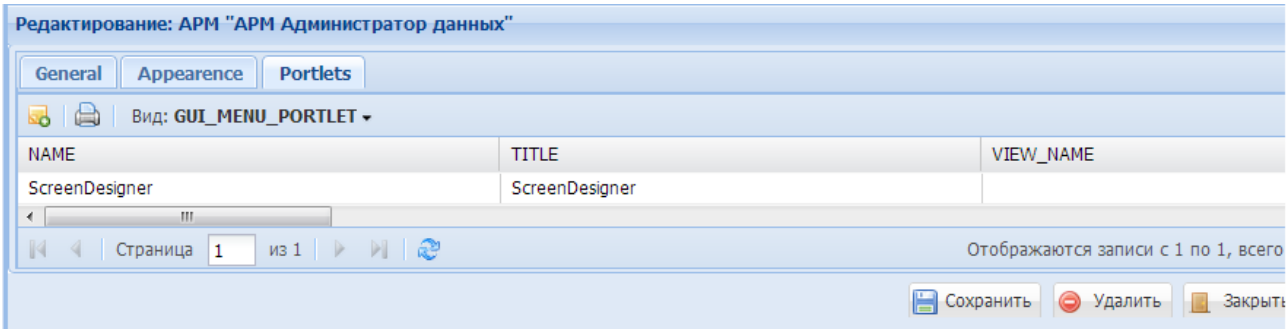


Рис. 5 Вкладка Portlets.

Вкладка Portlets активна только для АРМ.

Содержит список поддерживаемых портлетов конкретного АРМ.

Для использования стандартных представлений данных необходимо открыть вкладку «Функциональные модули» и перетащить выбранный модуль в меню.

7.4 Функциональные модули

Существует возможность использовать преднастроенные модули отображения данных - «Функциональные модули». Для этого мы создаем пункт меню, ссылающийся на тот или иной функциональный модуль. Есть два способа задания этой ссылки:

- создание ссылки вручную,
- перенос из списка функциональных модулей посредством drag & drop; При этом пункт меню создается автоматически,
- Вы можете ссылаться на раздел, состоящий из функциональных модулей. При этом меню системы определяет эту ссылку как «раздел меню».

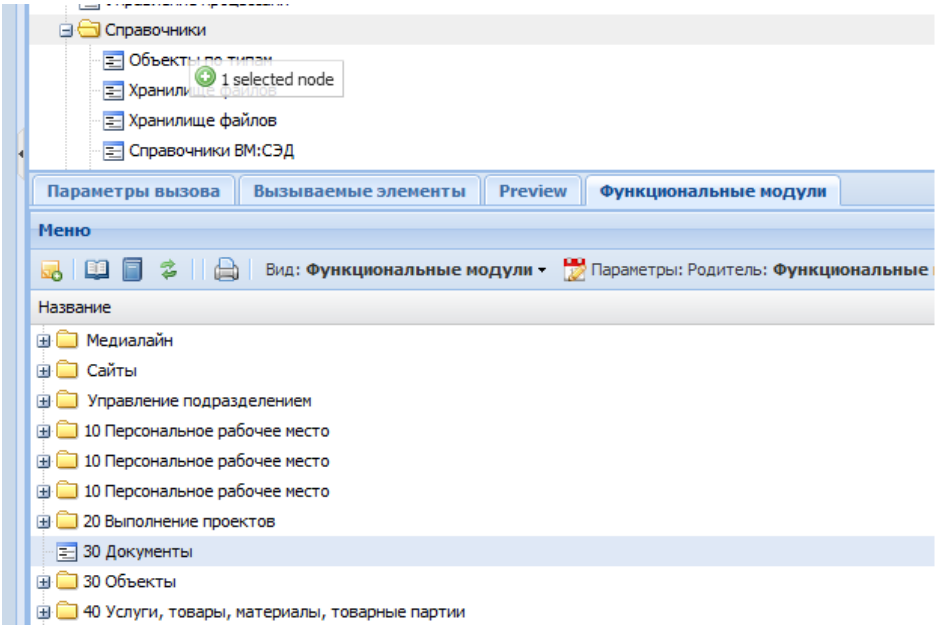


Рис. 6

Во вкладке Функциональные модули расположен список существующих функциональных модулей.

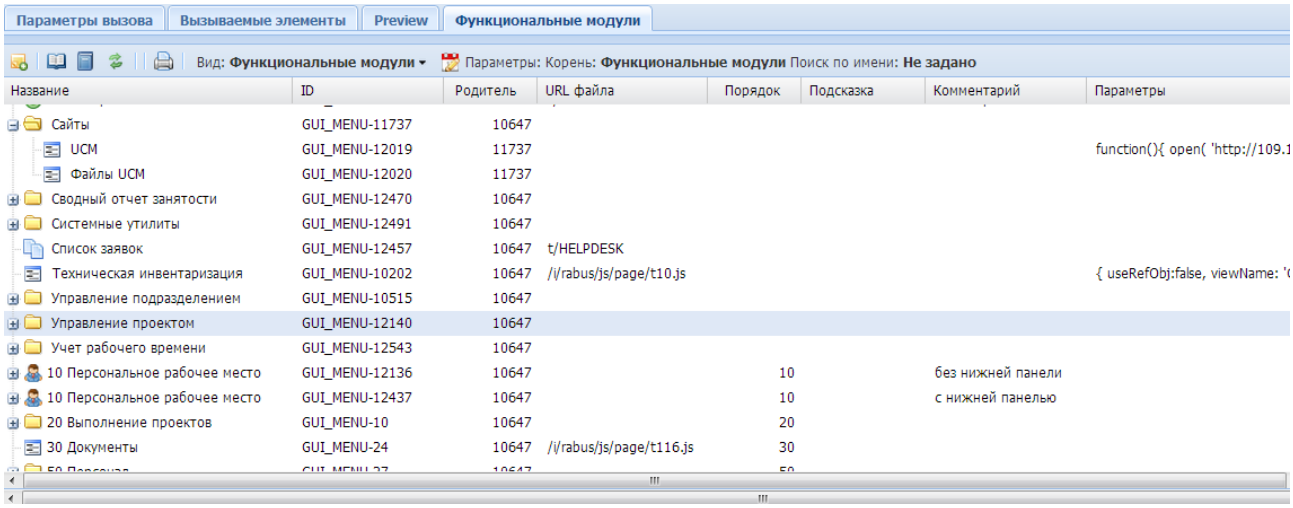


Рис. 7 Вкладка Функциональные модули

Для создания нового модуля необходимо нажать на кнопку «Создать новый объект» в левом верхнем углу данной вкладки.

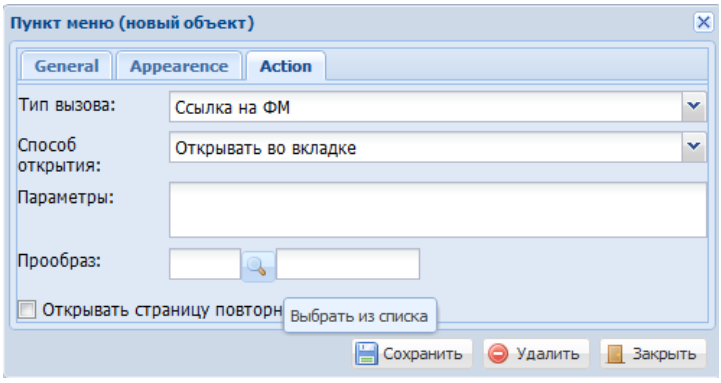


Рис. 8

Вкладка General.

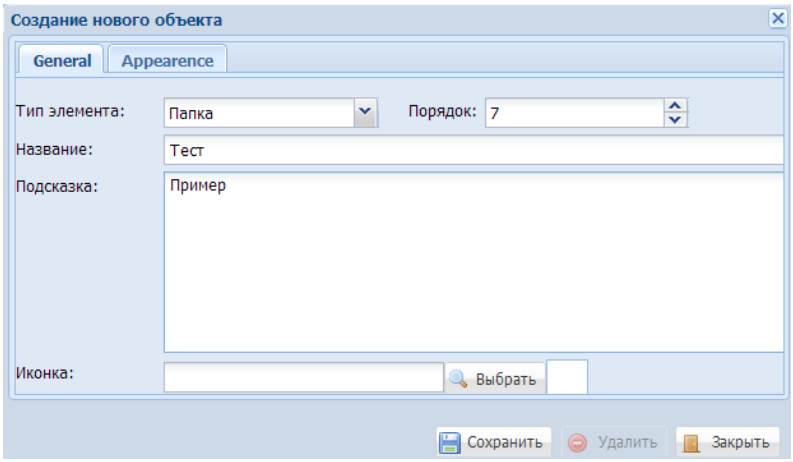


Рис. 9 Вкладка General.

Тип элемента. Позволяет выбрать тип добавляемого элемента: Папку или АРМ.

Порядок определяет положение элемента в списке. Если мы выберем пункт меню с уже существующим порядком и попытаемся создать новый элемент, то он будет создан рядом с выбранным пунктом, и его «Порядок» будет на единицу больше порядка соседнего элемента (в дереве элементов будет находиться на одну позицию ниже).

Название отображается в главное меню.

Иконка позволяет выбрать уже существующую иконку, либо загрузить новую.

Вкладка Appearence.

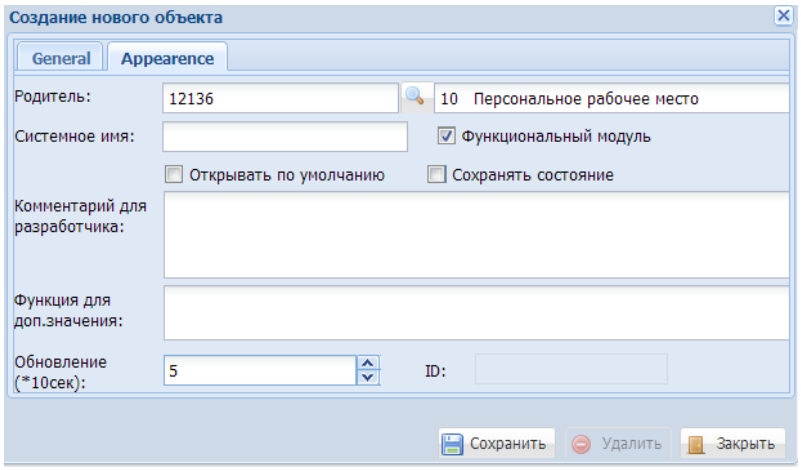


Рис. 10 Вкладка Appearence.

На этой вкладке мы видим служебные данные пункта меню.

Родитель: ID и имя родительского элемента.

Функциональный модуль: Флаг, определяющий «функциональный модуль»

Открывать по умолчанию. Если установлен флажок, то папка, АРМ, или раздел меню будут открыты сразу после загрузки главного меню, а пункт меню будет запущен в рабочей области.

Для изменения режима отображения конкретного функционального модуля необходимо кликнуть по вкладке «Вид» (рис. 11).

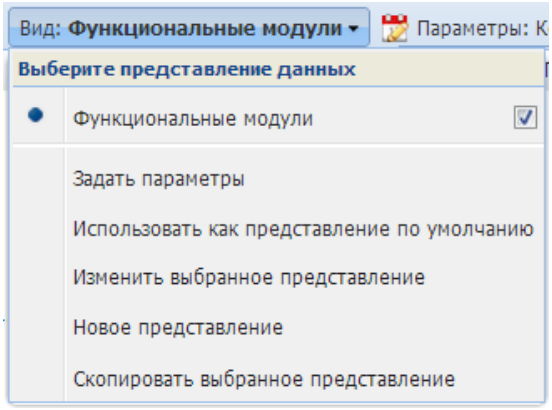


Рис. 11

Для изменения параметров представления необходимо выбрать пункт «Изменить выбранное представления», после чего откроется окно с параметрами представления.

Параметры панели «Представления» смотрите в справке по представлениям.

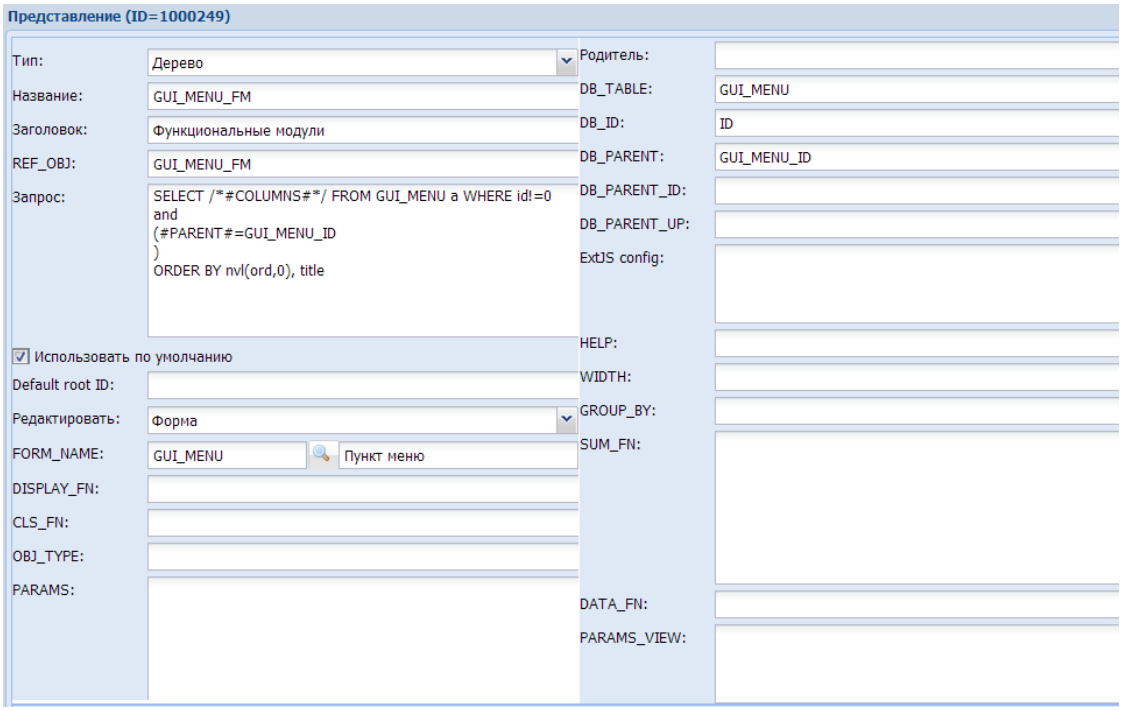


Рис. 12

8. Настройка Представлений

8.1 Общая информация

Что такое представление

Представление – Инструмент визуализации данных позволяющий менять интерфейс отображения данных, производить операции с имеющимся данными.

Другими словами представление – это инструмент ,необходимый для отображения нужных данных из БД, позволяющий менять интерфейс отображения, производить некоторые операции с имеющимися данными (например – сложение значений некоторых полей), корректировать данные. Главным преимуществом данного инструмента является то, что для работы с представлениями требуется невысокий уровень знаний в области работы с системой управления базами данных (СУБД). Это делает его доступными для более широкого круга людей.

Цели создания представлений

Многим пользователям, далеким от непосредственной работы с СУБД все же часто приходится сталкиваться с большим количеством данных, которые необходимо где-то хранить. Большинство современных программных комплексов используют различные БД, в качестве хранилища всей необходимой информации.

Для поддержания любой БД нужен администратор, умеющий с ней работать. Но для пользователей информацией все тонкости устройства хранения и редактирования данных не столь важны, поэтому встает вопрос о создании некоего инструмента, имеющего простой интерфейс и позволяющего быстро и качественно вывести на экран нужную информацию. Таким инструментом и является представление.

Возможности

Представления позволяет:

1. Отображать данные, находящиеся в БД в 3 видах
    1. в виде таблиц;

2.в виде дерева;

3.в виде интерактивного временного графика (TimeLine);

4.календаря;

5.графов.

2. Осуществлять выбор полей для отображения

3. Осуществлять вывод данных с определенным шаблоном

4. Осуществлять сортировку данных по столбцам

5. Осуществлять группировку данных по определенным полям

6. Осуществлять вывод аналитических данных

7. Определять форму редактирования, а так же возможность редактирования

8. Вызывать другие процедуры через панель инструментов

9. Определять и обрабатывать параметры ввода разных типов (строка, число, дата, справочные и пр.)

10. Печатать данное представление без дополнительной разработки отчетной формы

11. Сохранять значения представлений по умолчанию и для каждого пользователя

12. Силами администратора БД изменять вид страницы в связи с новыми требованиями Закона, стандартов Предприятия и требований Пользователя

Пользователи

Представлением пользуются:

1.Сотрудники, работающие с программным комплексом и выполняющие функциональные задачи – непосредственно Пользователи

2.Сотрудник(и) которые обслуживают базу данных и настраивают представления - Администраторы

8.2 Общий интерфейс

При работе с программным комплексом Пользователь видит результат работы представления

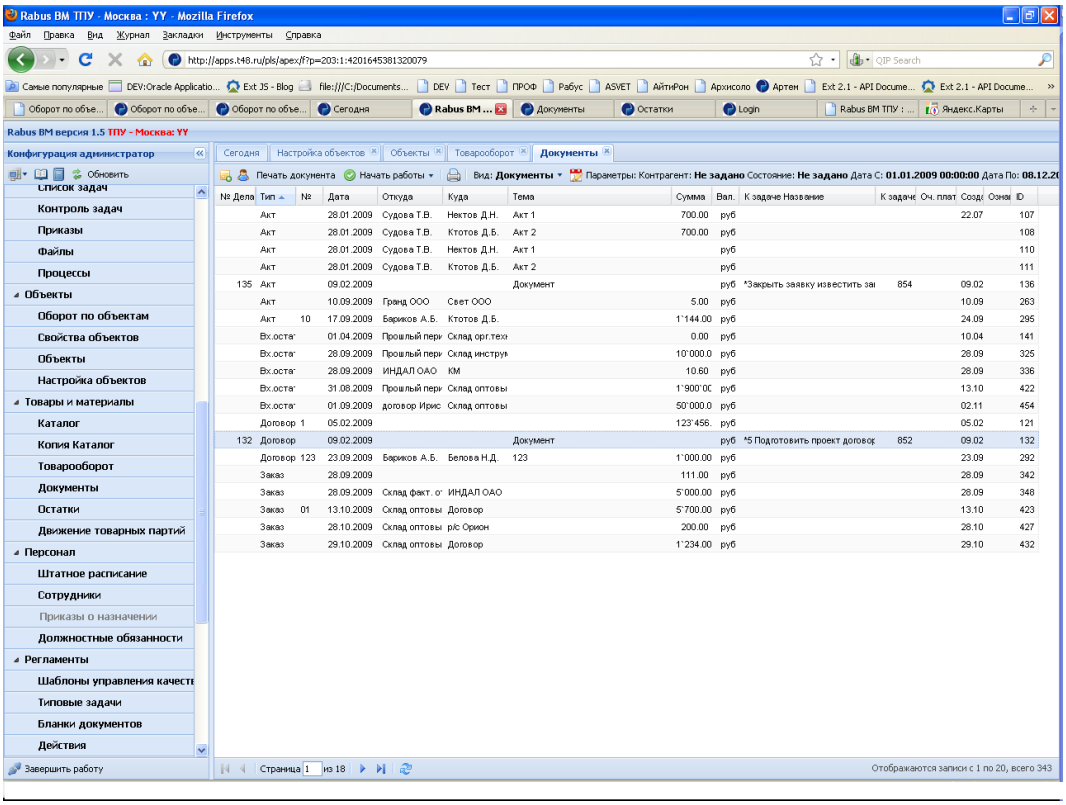


Рис. 1 Общий вид представления

Верхняя панель инструментов.

1) Кнопки доступные Пользователю

Кнопки представления – это стандартные кнопки, добавленные разработчиком на панель инструментов в зависимости от требований данного представления, находящиеся на верхней панели инструментов, до кнопки «Вид». Также кнопки могут быть созданы Администратором, в форме редактирование представления.

Рис. 2 Кнопки представления

2) Параметры

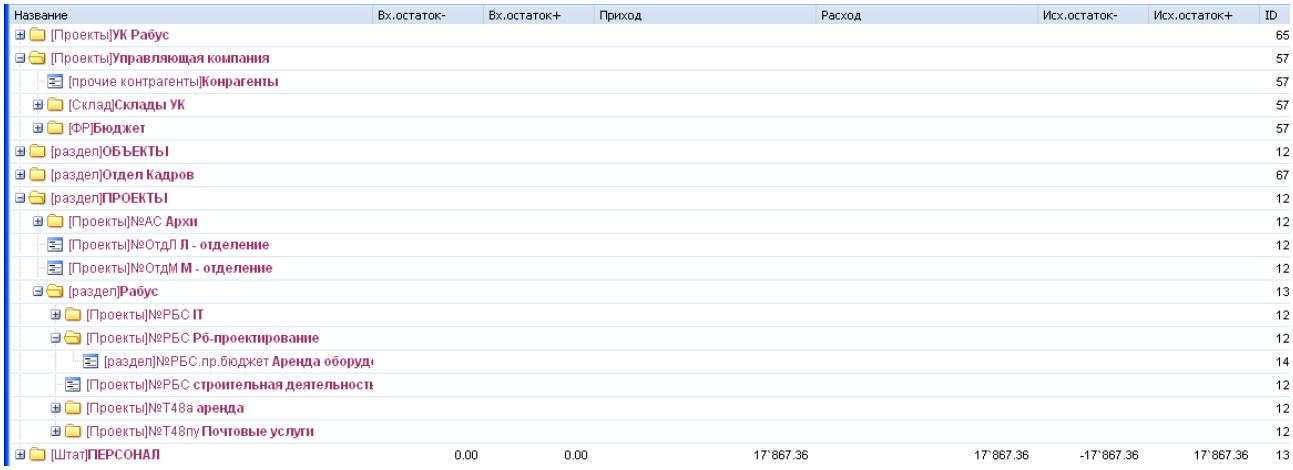
Параметры пишутся для каждого представления в зависимости от выполняемых представлением функций.

Рис. 3 Параметры представления

Отображение данных.

Данные в любом представлении находятся в рабочей области, и могут быть отображены в трех видах – в виде дерева, таблицы или в виде интерактивного временного графика (TimeLine.)

1) Отображение в виде дерева

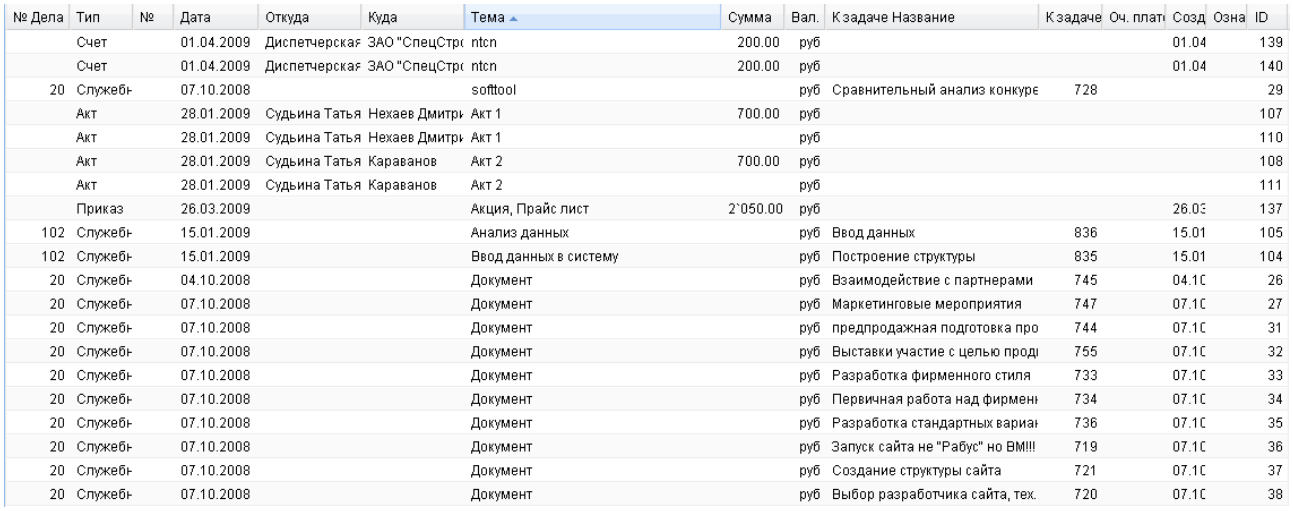


Название	Вх.остаток-	Вх.остаток+	Приход	Расход	Исх.остаток-	Исх.остаток+	ID
[Проекты]**УК Рабус**							65
[Проекты]**Управляющая компания**							57
[прочие контрагенты]**Конрагенты**							57
[Склад]**Склады УК**							57
[ФР]**Бюджет**							57
[раздел]**ОБЪЕКТЫ**							12
[раздел]**Отдел Кадров**							67
[раздел]**ПРОЕКТЫ**							12
[Проекты]№АС **Архи**							12
[Проекты]№ОтдЛ **Л - отделение**							12
[Проекты]№ОтдМ **М - отделение**							12
[раздел]**Рабус**							13
[Проекты]№РБС **IT**							12
[Проекты]№РБС **Рб-проектирование**							12
[раздел]№РБС.пр.бюджет **Аренда оборуд**							14
[Проекты]№РБС **строительная деятельност**							12
[Проекты]№Т48а **аренда**							12
[Проекты]№Т48пу **Почтовые услуги**							12
[Штат]**ПЕРСОНАЛ**	0.00	0.00	17`867.36	17`867.36	-17`867.36	17`867.36	13

Рис. 4 Вид представления – дерево

Сверху рабочей области находится строка с названиями колонок, заданных для отображения в данном представлении. Отображение в виде дерева имеет вид вложенных объектов слева и значения заданных колонок справа. Данное отображение позволяет быстро перемещаться по иерархической структуре.

2) Отображение в виде таблицы



№ Дела	Тип	№	Дата	Откуда	Куда	Тема ▲	Сумма	Вал.	К задаче Название	К задаче	Оч. плат	Созд	Озна	ID
	Счет		01.04.2009	Диспетчерская	ЗАО "СпецСтр	ntcn	200.00	руб				01.04		139
	Счет		01.04.2009	Диспетчерская	ЗАО "СпецСтр	ntcn	200.00	руб				01.04		140
20	Служебн		07.10.2008			softtool		руб	Сравнительный анализ конкуре	728				29
	Акт		28.01.2009	Судьина Татья	Нехаев Дмитри	Акт 1	700.00	руб						107
	Акт		28.01.2009	Судьина Татья	Нехаев Дмитри	Акт 1		руб						110
	Акт		28.01.2009	Судьина Татья	Караванов	Акт 2	700.00	руб						108
	Акт		28.01.2009	Судьина Татья	Караванов	Акт 2		руб						111
	Приказ		26.03.2009			Акция, Прайс лист	2`050.00	руб				26.03		137
102	Служебн		15.01.2009			Анализ данных		руб	Ввод данных	836		15.01		105
102	Служебн		15.01.2009			Ввод данных в систему		руб	Построение структуры	835		15.01		104
20	Служебн		04.10.2008			Документ		руб	Взаимодействие с партнерами	745		04.10		26
20	Служебн		07.10.2008			Документ		руб	Маркетинговые мероприятия	747		07.10		27
20	Служебн		07.10.2008			Документ		руб	предпродажная подготовка про	744		07.10		31
20	Служебн		07.10.2008			Документ		руб	Выставки участие с целью проди	755		07.10		32
20	Служебн		07.10.2008			Документ		руб	Разработка фирменного стиля	733		07.10		33
20	Служебн		07.10.2008			Документ		руб	Первичная работа над фирмен	734		07.10		34
20	Служебн		07.10.2008			Документ		руб	Разработка стандартных вариан	736		07.10		35
20	Служебн		07.10.2008			Документ		руб	Запуск сайта не "Рабус" но ВМ!!!	719		07.10		36
20	Служебн		07.10.2008			Документ		руб	Создание структуры сайта	721		07.10		37
20	Служебн		07.10.2008			Документ		руб	Выбор разработчика сайта, тех.	720		07.10		38

Рис. 5 Вид представления – таблица

Сверху рабочей области находится строка с названиями колонок, заданных для отображения. Данные представлены в виде таблицы. Этот тип отображения данных позволяет производить сортировку по какой либо колонке,

путем нажатия мышки на определенной колонке, или же, рядом с каждой колонкой есть выпадающее меню – в нем можно выбрать вид сортировки. Также можно группировать элементы по полям (данный параметр находится в том же выпадающем списке – пункт меню «Группировать по данному полю»), производить выборку данных (указав нужные параметры в пункте выпадающего меню «Фильтр»), отображать или скрывать столбцы представления (в пункте выпадающего меню «столбцы» галочки стоят напротив отображаемых столбцов – если их убрать, колонка будет скрыта).

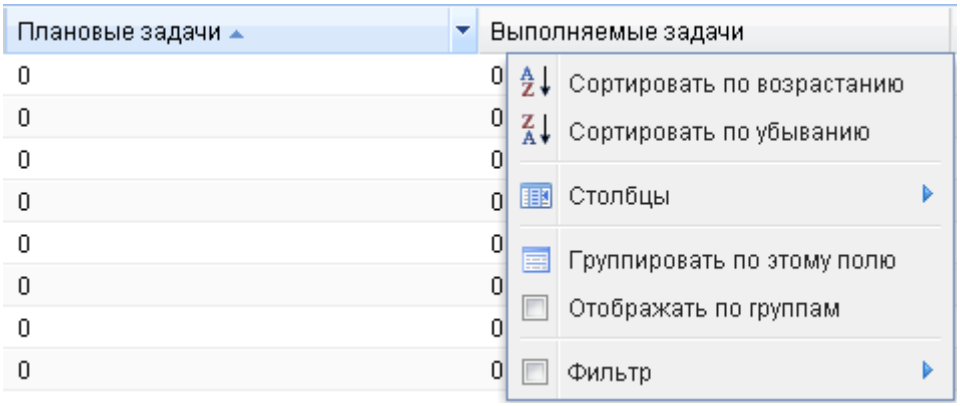


Рис. 6 Сортировка

Нижняя панель инструментов представляет собой полосу с навигацией по таблице, также на ней находится общая информация о таблице – общее количество записей и количество отображаемых в данный момент.

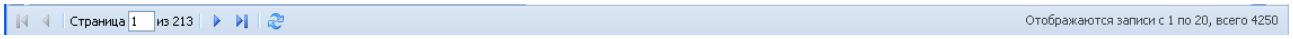


Рис. 7 Нижняя панель

3) Отображение в виде интерактивного временного графика (TimeLine)

Рис. 8 Отображение представления – TimeLine

Отображение в виде TimeLine – график задач. Красные флажки – список документов, соответствующий конкретной задаче. Красная полоса обозначает текущую дату. Данный график строится по временным меткам – дата начала выполнения задачи, дата окончания. У каждой задачи есть 4 даты: плановая дата начала, плановая дата окончания, фактическая дата начала, фактическая дата окончания. При отображении на временном графике диапазоны между указанными датами обозначаются разным цветом. Пользователь может выбрать свой вариант цветовой схемы для задач при помощи соответствующей кнопки в панели инструментов.

Формирование данных для представления timeline аналогично формированию данных для типа grid.

Особенностью timeline является фиксированные имена колонок:

ID - ID задачи

PLAN_FD — плановая дата начала

PLAN_TD — плановая дата завершения

FACT_FD — фактическая дата начала

FACT_TD — фактическая дата завершения

SUBJ — надпись на полоске задачи

Для того, чтобы документы, относящиеся к задаче отображались в виде красных флажков расставленых над задачей, в описании представления в поле DATA_FN нужно написать helper_task.docs_for_task(:1,:2)

8.3 Действия с представлениями

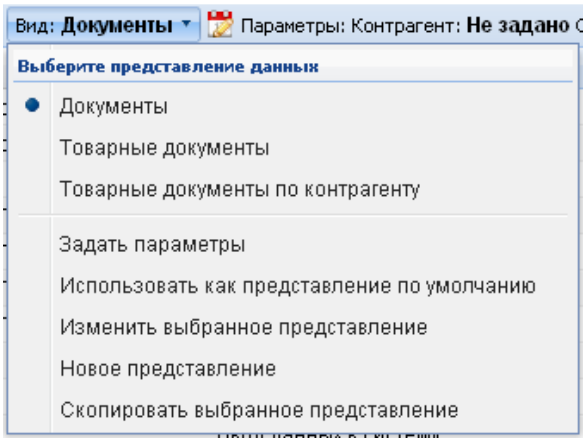


Рис. 9 Меню представления

1) Создание представления

Для создания представления необходимо открыть выпадающий список кнопки «Вид» и выбрать пункт меню «Новое представление», после чего откроется форма редактирования представления (рис. 10).

2) ID представления

Формируется в момент сохранения представления. Оно уникально для любого представления системы.

3) Обязательные для заполнения поля

Обязательными полями для создания представления являются поля Тип, Название, Заголовок и REF_OBJ. Остальные поля вы можете заполнить позже.

4) Изменение представления

Для изменения представления необходимо раскрыть выпадающий список кнопки «Вид» (рис. 9) и выбрать пункт меню «Изменить выбранное представление», после чего появится форма для редактирования представления.

5) Копирование представления

Копирование представления позволяет создать идентичное представление. Для этого нужно открыть выпадающий список кнопки «Вид» (рис. 9) и выбрать пункт меню «Скопировать выбранное представление». После нажатия данной кнопки меню вам необходимо обновить страницу, и в списке представлений вы найдете скопированное. Имя у нового представления формируется автоматически. При копировании представление сохраняет все настройки, созданные параметры, поля, за исключением кнопок.

6) Задание параметров представления

Для задания параметров выбранного представления, необходимо открыть выпадающий список меню кнопки «Вид» (рис. 9) и выбрать пункт «Задать параметры» или нажать на кнопку «Параметры», правее кнопки «Вид». Появится окошко, в котором следует указать нужные значения существующих параметров. Если параметров у представления нет, то появится окошко с сообщением «У выбранного представления нет параметров».

7) Использование представления по умолчанию

Использовать представления по умолчанию означает, что при загрузке страницы будет видно данное представление. Для выбора такого свойства представления необходимо открыть выпадающий список меню кнопки «Вид» и выбрать пункт «Использовать как представление по умолчанию» или же поставить галочку в форме редактирования представления в пункте «Использовать по умолчанию».

8) Удаление представления

Для удаления представления необходимо зайти в форму для редактирования представления (выпадающий список кнопки «Вид», далее «Изменить выбранное представление») и нажать кнопку удалить.

8.4 Форма редактирования представления

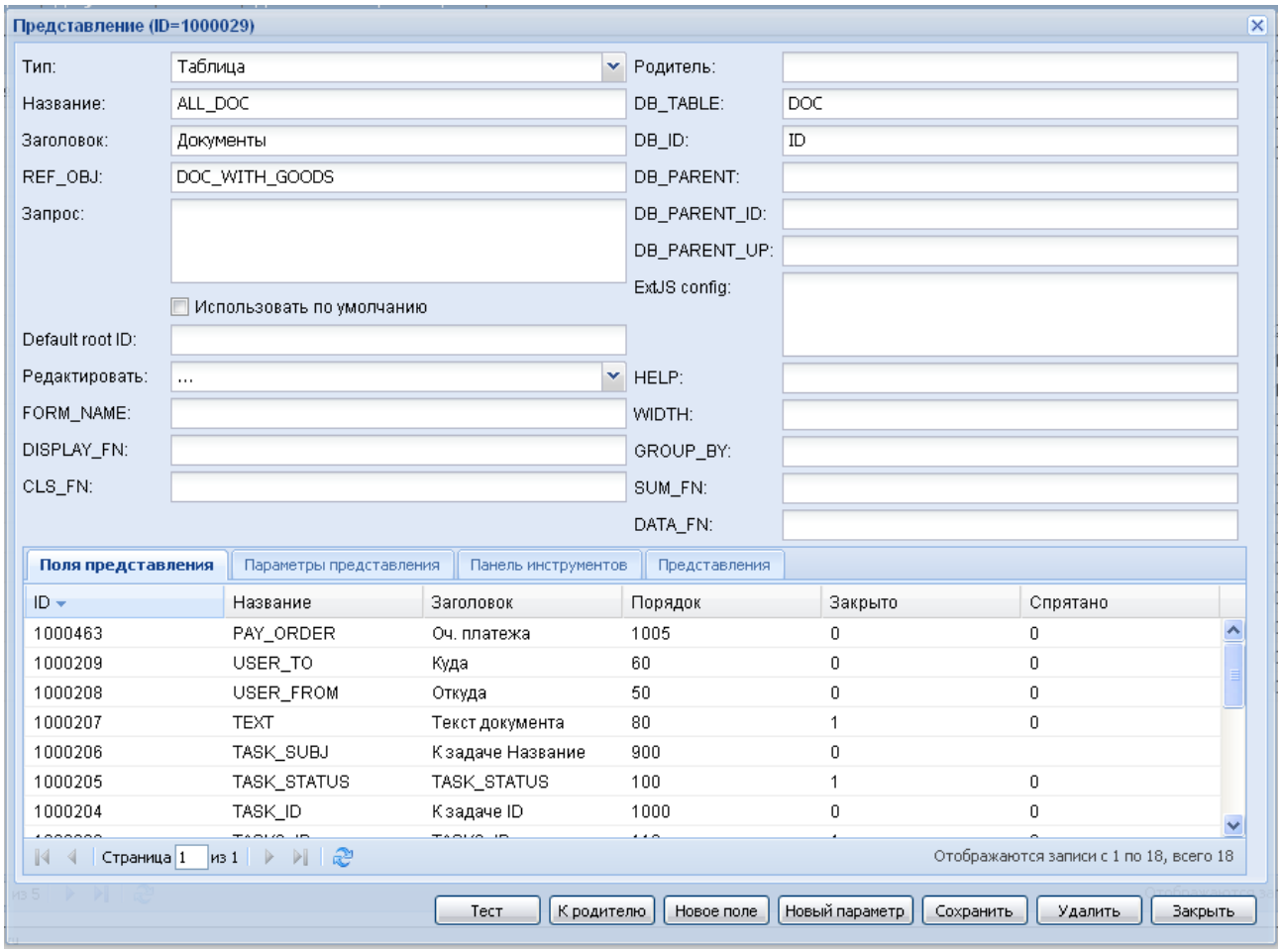


Рис. 10 Форма редактирования представления

1) Поля представления

Поле Тип. В поле «Тип» вы можете выбрать тип отображения данных, таких как Таблица, Дерево, TimeLine.

Поле Название. В поле название вы пишете название представления, которое в дальнейшем будет использоваться системой. Название должно быть уникально.

Поле Заголовок. В данном поле вы пишете заголовок представления – данное название представление, которое будет отображаться на странице.

Поле REF_OBJ. Данное поле отвечает за группировку представлений между собой – при указании данного параметра следует посмотреть значение этого поля у других представлений, которые расположены на той странице, где вы хотите создать свое. Все представления с одинаковыми значениями данного поля образуют набор, привязанный к определенной странице и видимый в выпадающем списке кнопки «Вид» (Для выбора страницы, к которой вы хотите привязать свое представление, следует помнить, что все представления на одной странице должны быть одинакового типа).

Ниже расположена строка «Использовать по умолчанию». Если вы хотите, что бы данное представление в данном наборе отображалось по умолчанию, поставьте галочку у этой строки.

Поле Запрос. Данное поле заполняется только в том случае, если тип представления – дерево. В поле пишется шаблон запроса на языке SQL. Для остальных типов представлений, запрос формируется исходя из других параметров.

Поле Default root ID. Если тип отображения данных – дерево, то в данном поле пишется ID корневого элемента дерева. Если поле пусто, то это эквивалентно надписи 0. В данном поле возможны только числовые значения (0,1,2 и т.д.)

Поле Редактировать. Данное поле используется в паре со следующим полем «FORM_NAME».

Данное поле имеет выпадающий список значений.

Пункт «Построчно» (доступен только в представлениях - таблицах) позволяет в дальнейшем редактировать записи представления по одинарному клику, непосредственно в рабочей области. Для этого в поле «FORM_NAME» необходимо указать форму с тем же набором полей, которые будет нужно редактировать.

Пункт «Форма» позволяет осуществлять редактирование в открывшейся форме, в поле «FORM_NAME» необходимо указать имя этой формы.

Пункт «Запрещено» не позволяет редактировать записи в представлении.

Поле FORM_NAME. Данное поле используется в паре с полем «Редактировать». В поле указывается имя формы для выполнения функций поля «Редактировать».

Поле DISPLAY_FN. Данное поле используется для представлений, которые вызываются при задании параметра rbs_lookup, он выглядит это как строка, разделенная на 3 части – активная область, картинка с лупой и закрытая

область. В активную область как раз заноситься ID элемента, в закрытой области выводится преобразованное значение ID в буквенном эквиваленте.

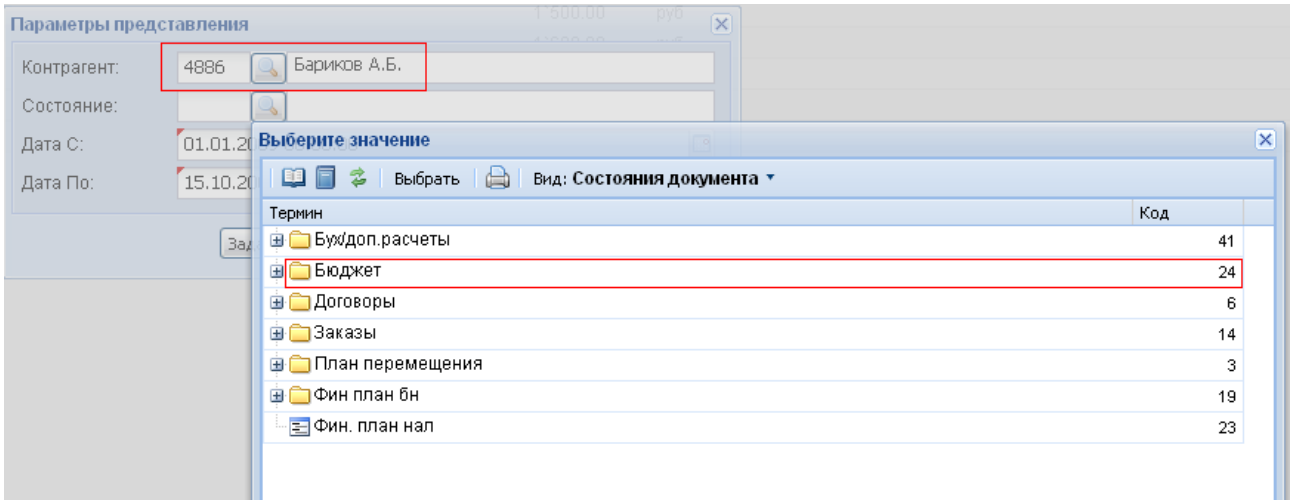


Рис. 11 Представление вызвано параметром rbs_lookup

На рис. 11 представление Состояние документа вызвано параметром «Состояние», а параметр «Контрагент» уже задан, в правой части отображается «смысл» ID заданного слева.

В поле DISPLAY_FN записывается функция формата «utl_doc.get_name(:1)», в переменную :1 далее будет передан ID выбранного элемента.

Поле CLS_FN. В данном поле нужно вызвать функцию, формирующую некоторый кусок CSS таблицы, значения которой будут использоваться данным представлением. Для использования данного поля в представлении обязательно должна быть колонка CLS.

Поле Родитель. Данное поле заполняется если представление является дочерним, в него пишется ID родительского представления.

Поле DB_TABLE. В данном поле пишется имя основной для данного представления таблицы.

Поле DB_ID. В поле DB_ID пишется название поля, которое уникально (чаще всего ID) для основной для представления таблицы.

Поле DB_PARENT. В данном поле пишется имя поля таблицы, являющейся ссылкой на запись родительского элемента . поле заполняется для представления – дерева,.

Поле DB_PARENT_ID. Если в представлении типа дерево разрешено перетаскивание объектов, то в данном поле пишется название поля таблицы, по которому можно узнать ID родительского объекта.

Поле DB_PARENT_UP. В данное поле следует поместить sql-выражение, которое позволяет определить значение родительского ID (используется при формировании данных для добавления нужной вершины в представление)

Поле GROUP_UP. В данном поле пишутся имена колонок, по которым нужно сгруппировать отображаемые данные.

Поле SUM_FN. Если представлению необходимо суммирование по колонкам, одной колонке или какие-либо еще математические операции с данными, а данном поле можно указать функцию, которая почив заданные пользователем параметры представления вернет требуемый ответ.

Например: view_obj.get_sum_column(:1). Вместо :1 в функцию подставляется объект, сформированный из всех параметров, указанных пользователем и системных.

Поле DATA_FN. Позволяет вызвать функцию, которая вызывается для каждой строки данных, полученных в качестве ответа с сервера, и преобразовать эти данные в любой другой вид, по желанию программиста.

Поле PARAMS. Содержит объект javascript, поля которого будут добавлены к объекту config, задающему свойства представления по умолчанию. Значения свойств объекта PARAMS имеют приоритет по отношению к свойствам объекта config.

Следующие свойства объекта PARAMS обрабатываются специальным способом:

cell_edit – true включить редактирование данных в ячейках (для Ctree)

raw_id – true означает что в идентификаторах вершин дерева отсутствует префикс вида ИМЯ_ТАБЛИЦЫ- (для Ctree)

store_config – применяется к конфигу для объекта Ext.data.GroupingStore через Ext.apply

page_size – задает количество выводимых на одной странице Grid свойств

on_ready – произвольная javascript функция, выполняется при инициализации представления

Поле PARAMS_VIEW. Содержит объект javascript, поля которого будут добавлены к объекту config, задающему свойства служебного объекта Ext.grid.GroupingView, используемого при работе представления. по умолчанию. Значения свойств объекта PARAMS_VIEW имеют приоритет по отношению к свойствам объекта config.

Следующие свойства объекта PARAMS_VIEW обрабатываются специальным способом:

getRowClass – javascript функция, позволяет вычислять css для всей строки представления.

Остальные поля. Поля «ExtJS config», «HELP», «WIDTH» зарезервированы для будущей функциональности.

2) Вкладка «Поля представления»

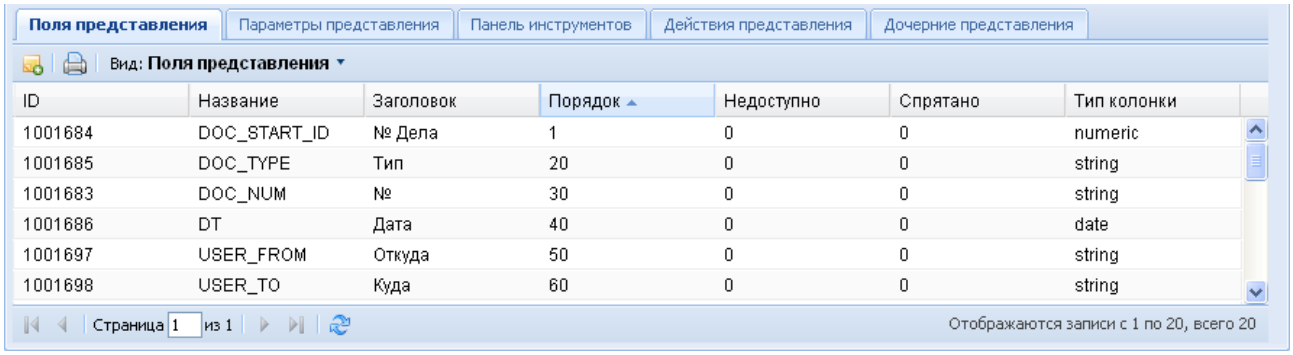


Рис .12 Поля представления

Вкладка «Поля представления» представляет собой представление - таблицу, данные в которой являются списком полей редактируемого представления. Сверху видна строка с перечнем свойств столбцов. Это стандартные свойства, характерные для описания любой колонки представления. Снизу вкладки есть навигация, а также общая информация о таблице – количество записей и какие отображаются в данный момент. В левом углу кнопка позволяет добавить новое поле.

3) Вкладка «Параметры представления»

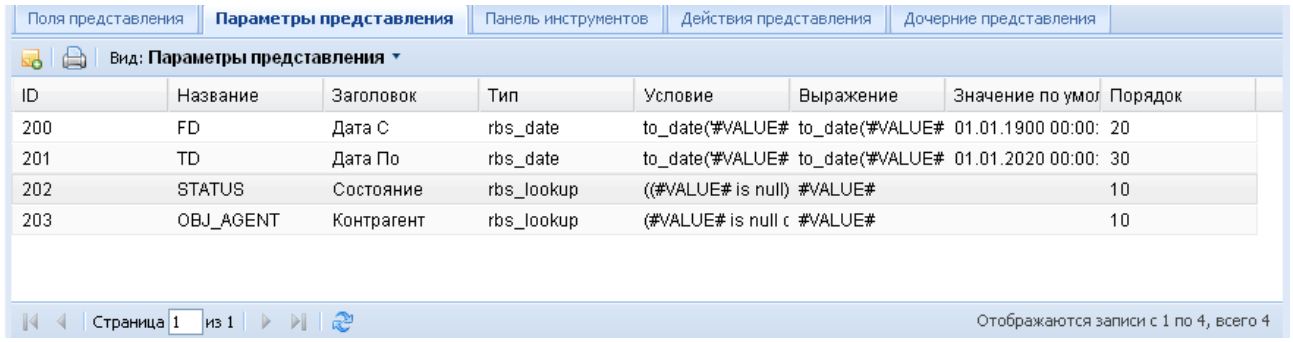


Рис. 13 Параметры представления

Вкладка «Параметры представления» представляет собой представление - таблицу с перечнем параметров, используемых для работы данного представления. Сверху видна полоса с названиями стандартных свойств параметров, внизу – навигация по таблице и общая информация о записях.

Первая кнопка слева позволяет добавить новый параметр.

4) Вкладка «Панель инструментов»

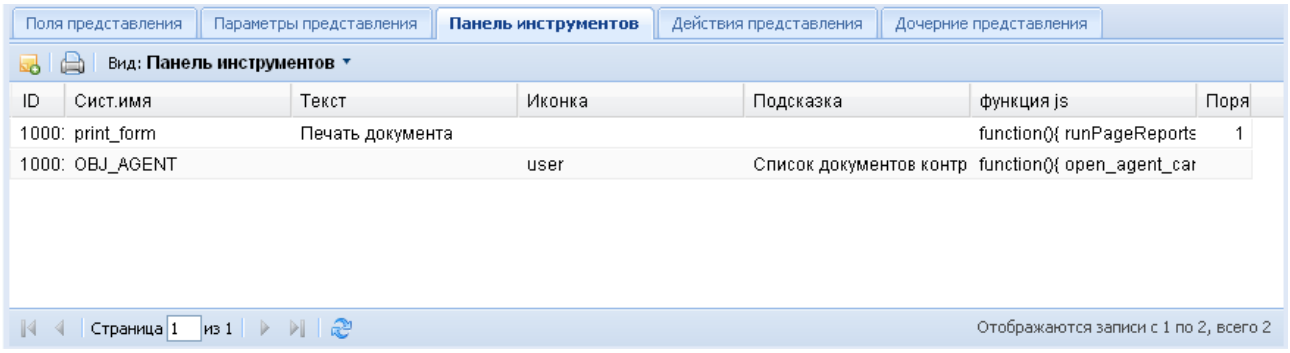


Рис. 14 Панель инструментов

Вкладка «панель инструментов» представляет собой представление - таблицу. Данные этой вкладки – кнопки, видимые пользователем в разделе «Кнопки представления».

5) Вкладка «Дочерние представления»

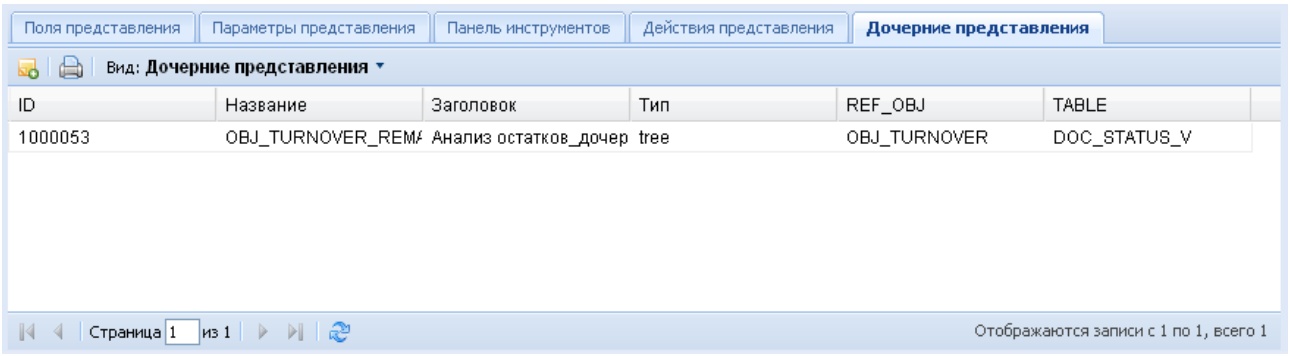


Рис. 15 Дочерние представления

Вкладка «представления» также является представлением - таблицуй, где описываются все дочерние представления выбранного. При двойном щелчке на любом из пунктов таблицы откроется форма редактирования соответствующего дочернего представления.

6) Кнопки формы

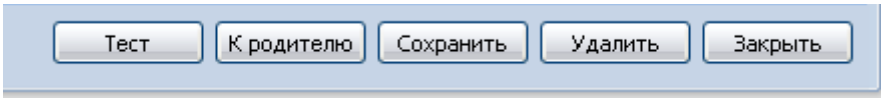


Рис. 16 Кнопки формы

Кнопка Тест. Позволяет сгенерировать тестовый вариант представления с заданными настройками.

Кнопка к родителю. Кнопка «К родителю» позволяет открыть форму редактирования родительского представления.

Кнопка Сохранить. Сохраняет все свойства формы редактирования для представления.

Кнопка Удалить. Удаляет выбранное представление.

Кнопка Закрыть. Закрывает форму редактирования представления.

1.5 Создание полей

Если представление создано на основе таблицы, в которой отсутствует поле ID то решением проблемы будут следующие действия:

Если в таблице есть поле, являющиеся ключом, то в поле DB_ID следует указать его.

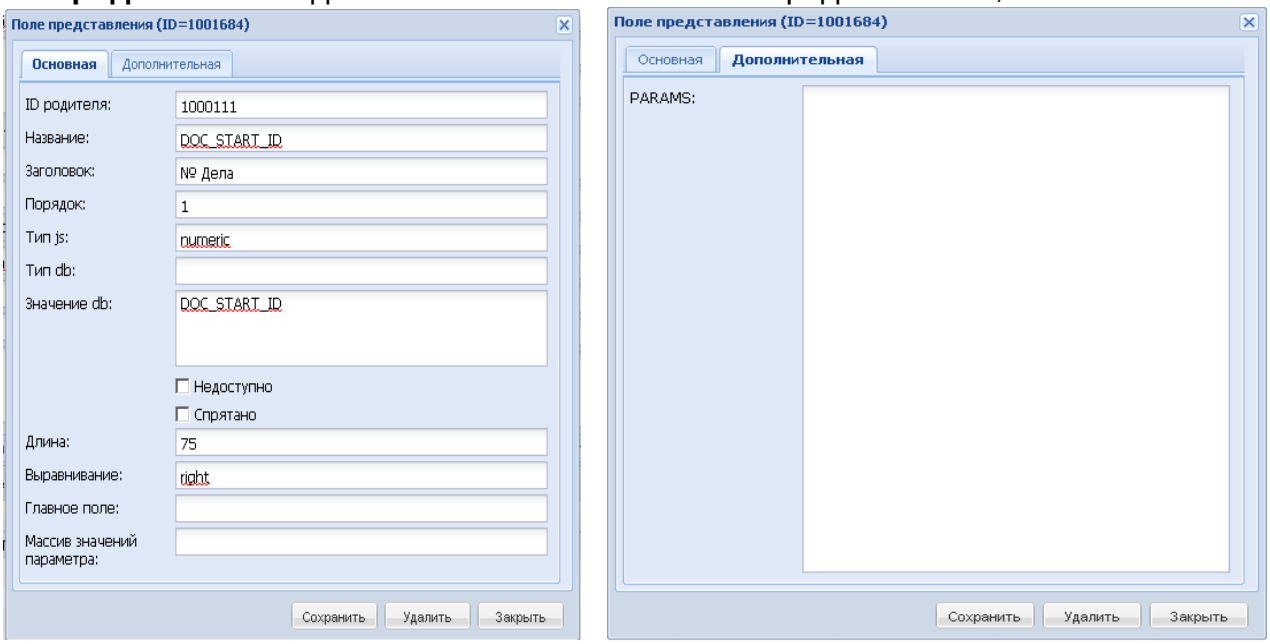
Если необходимо чтобы представление в форму редактирования передавало ключ состоящий из нескольких полей, нужность указать в поле OBJ_TYPE значение SYSNAME того типа объекта, который содержит все те поля как и таблица, на основе которой мы строим представление. Для свойств этого типа объекта необходимо отметить те, которые являются частью составного ключа.

1) Форма «Поле представления»

Для создания нового поля нужно открыть форму редактирования представления и нажать на кнопку в левом верхнем углу вкладки «Поля представления», при желании редактировать уже существующее поле необходимо два раза щелкнуть на строке с нужным полем.

Рис. 17 Форма - поле представления

Поле ID родителя. В данном поле пишется ID представления, поле заполняется автоматически.



Поле Название. В данном поле пишется системное имя поля (Латиница).

Поле Заголовок. В этом поле пишется тот текст, который будет отображен на странице представления.

Если колонки формируются динамически, то имя колонки так же можно сформировать динамически.

Пример: 'Приход с #ARRAY_VAL#', в этом случае вместо #ARRAY_VAL# подставится очередное значение массива, сформированного из поля «Массив значений»

Поле Порядок. Пишется номер, который влияет на расположение колонки. Т.е. это порядковый номер колонки в представлении.

Поле тип JS. Отвечает за тип данных, которые будут отображены в данной колонке. Возможны варианты – String, numeric, date, boolen, пустой тип – соответственно – строковый тип, числовой, тип дата, логический тип и пустой тип. Пустой тип позволяет хранить в этой колонке любой тип данных, например картинки, документы и т.д.

Тип db. Зарезервировано для будущей функциональности.

Поле Значение db. Значение, которое будет отображаться в данной колонке – это может быть либо значение поля в таблице, либо имя функции (с параметрами), либо запрос вида (select …….), скобки обязательны.

Если в вызываемую функцию нужно передать параметры, то пишется:

/*#<имя параметра>_VAL#*/ - это значит, что в это место передастсязначение параметра, выбранного пользователем. Причем если пользователь еще ничего не выбрал, ошибки не будет.

В случае если нужно передать имя поля, то просто пишется его имя.

Пример:

helper_obj_turnover.get_enter_amount_obj(/*#FD_VAL#*/,id,/*#CURR_VAL#*/,/*#STATUS_VAL#*/,'false')

В примере id-имя поля в таблице БД, на место /*#FD_VAL#*/, будет передано выбранное значение даты.

Поле Недоступно. Может иметь два значение – 0 и 1 – соответственно 0 – колонка будет доступна, 1- недоступна. Если колонка недоступна, то она не обрабатывается программой.

Поле Спрятано. Может иметь два значение – 0 и 1. соответственно 0 – поле не спрятано, 1- спрятано. При этом колонка обрабатывается программой, просто не отображается в представлении.

Поле Длина. В данном поле пишется длина поля в пикселях. Причем если тип отображения – дерево и суммарная длина всех полей меньше ширины экрана, то весь оставшийся запас пойдет на колонку NAME. Если больше ширины экрана, то все длины пропорционально сожмутся. Если же не задавать это поле, то выставятся средние значения.

Поле Выравнивание. Возможные значение left – right соответственно это выравнивание данных по левому и правому краю ячейки.

Поле Главное поле. Поле, в котором строится дерево, если тип отображения – дерево. При этом в данное поле пишется цифра 1.

Поле «Массив значений». В данном поле описывается массив значений для случая, когда колонки формируются динамически.

Пример синтаксиса ДИАПАЗОН #FD# #PERIOD# #TD#, в данном случае FD, PERIOD,TD – параметры данного представления. При такой записи сформируется массив дат с шагом равным значению параметра PERIOD.

Поле «PARAMS». Данное поле служит для форматирования колонки представления. Значением поля должна являться строка, которая после преобразования функцией js может быть применена к модели колонки.

Следующие свойства объекта PARAMS обрабатываются специальным способом:

renderType – задает форматирование значений

'datetime' 'date': дата, формат задается значением dateFormat

'money' – выравнивание по правому краю, формат денежный

dateFormat – формат даты, в котором представлению поступают данные типа Дата для колонк

renderer – задает js функцию, которая будет создавать оформление ячейки

Пример 1:{ width: 60, menuDisabled: True}

Колонка с заполненным полем PARAMS таким образом будем иметь ширину 60 px и неактивное выпадающее меню.

Пример 2: {

renderer: function(value,metaData){

var attr = metaData.attr || '';

metaData.attr =attr + ' style="font-style:italic;"';

return value;

}}

Этот пример показывает как можно изменить шрифт значения ячеек в колонке.

2) Зарезервированные имена колонок

Для возможности форматирования (выделения) колонок и строк было создано два специальных типа поля представления. Это поля ИМЯ_ПОЛЯ_STYLE_ и ROW_CLASS_

Если мы хотим выделить строку в представлении необходимо использовать поле ROW_CLASS_.

Пример: Необходимо в представлении документы строки с типом документа 1 подкрасить зеленым, а если у документа сумма нулевая то красным.

В этом случае создаем колонку представления ROW_CLASS_

Заполняем у данной колонки поле «Значение db» , делаем колонку спрятанной.

Значение поля «Значение db»

CASE

WHEN doc_type =1 THEN 'green-row'

WHEN amount is null then 'red-row'

END

Этот кусок кода на языке sql добавится к запросу при формировании представления.

В данном примере 'green-row' и 'red-row' это заранее определенные классы форматирования описанные в css файле. Так же необходимо чтобы кроме классов 'green-row' и 'red-row' были определены классы «red-row-selected» и «green-row-selected». Классы …-selected служат для подсветки выбранной строки.

Если необходимо форматировать данные в колонке, тогда необходимо использовать ИМЯ_КОЛОНКИ_STYLE_

Пример: Необходимо выделить красным цветом те значения состояний документов, в которых суммы равны нулю.

Создаем колонку STATUS_STYLE_ (т. к. имя колонки Состояние - STATUS)

Заполняем у данной колонки поле «Значение db» , делаем колонку спрятанной.

Значение поля «Значение db»

CASE

WHEN amount is null THEN 'color:’red''

END

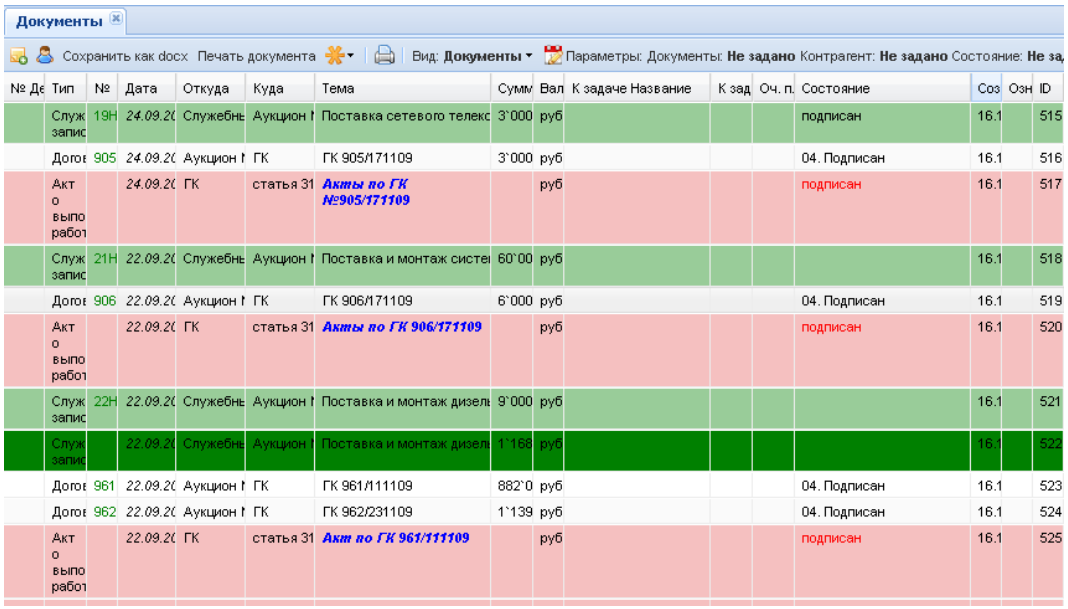


Рис. 18 – Форматированное представление

8.5 Создание параметров

1) Форма «Параметр представления»

Для создания параметра, необходимо открыть форму редактирования представления.

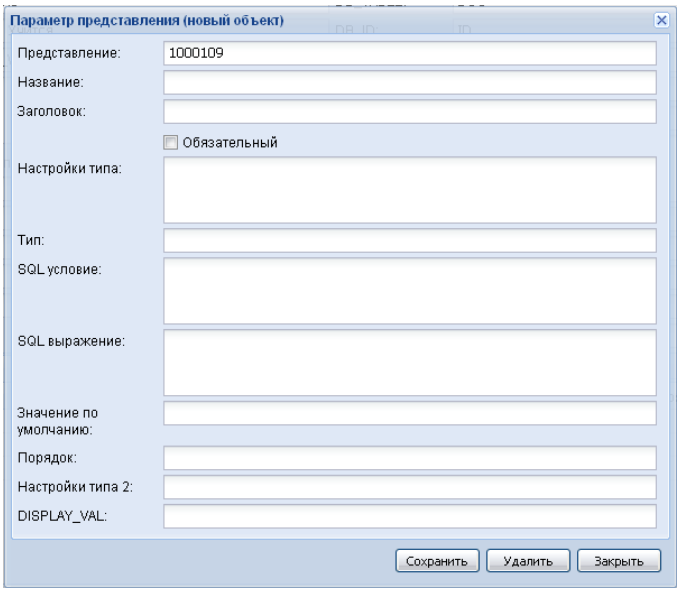


Рис. 19 Форма - Параметр представления

Поле Представление. В данном поле пишется ID представления, для которого создается параметр и заполняется это поле автоматически.

Поле Название. Системное имя параметра, необходимое для программного обращения к параметру.

Поле Заголовок. Название параметра, которое будет отражено на панели инструментов представления. Если поле заголовок пусто, то параметр будет невидимым.

Поле Тип параметра. Определяет отображение параметров при их задании (табл1).

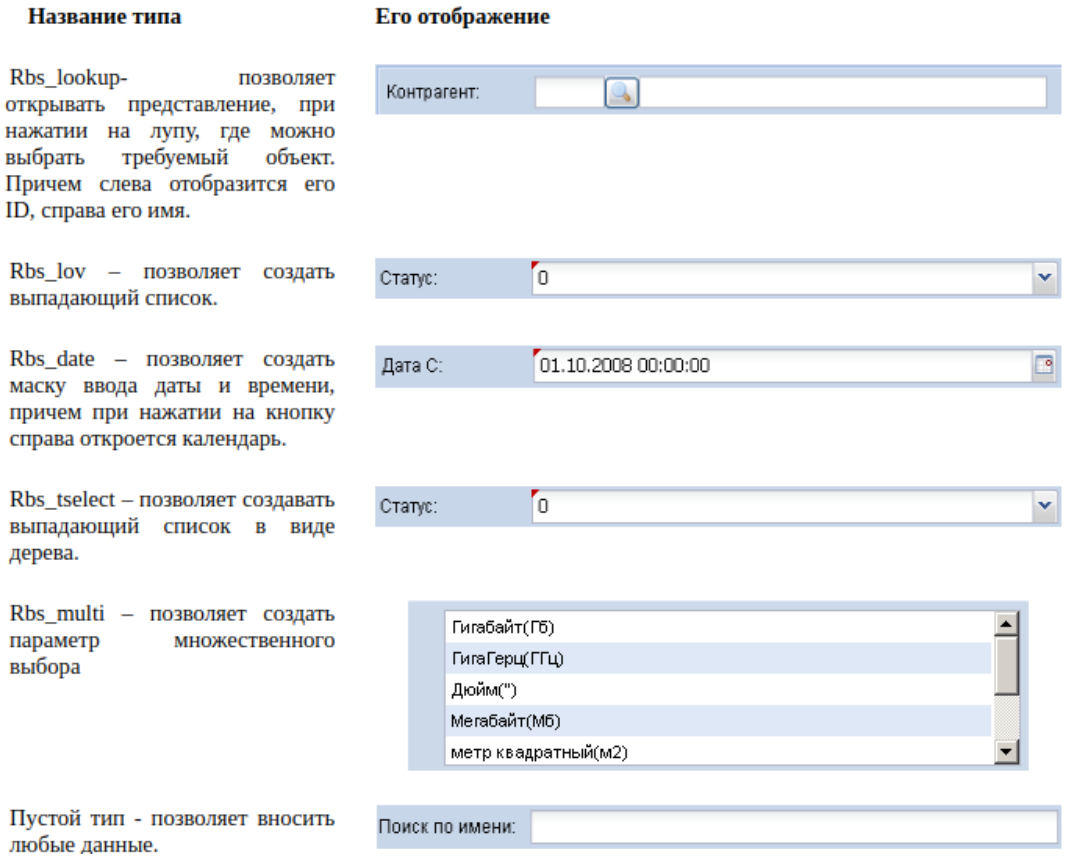


Рис. 20 Типы параметров

Поле Настройка типа. Определяется в зависимости от типа параметра (Таблица 1).

Таблица 1 - Настройка типа параметра

Название типа	Настройка типа
Rbs_lookup	Пишется имя представления, которое будет появляться при нажатии на лупу

Rbs_lov	Пишется имя списка из таблицы GUI_LOV (значения данного списка будут составлять выпадающее меню)
Rbs_date	Пусто
Rbs_tselect	Пишется имя отображаемого дерева
Rbs_multi	Пишется имя списка из таблицы GUI_LOV (значения данного списка будут составлять выпадающее меню)
Пусто	Пусто

Поле SQL-условие. Пишется условие, на языке SQL, но без слова WHERE, в круглых скобках. Если параметр может быть пустым, то нужно уточнить – либо ничего не выводить в данном случае, либо выводить все. Для вывода всего, в случае пустого значения параметра пишется:

((‘#VALUE#’ is null) or ...)

Для типа параметра rbs_multi это поле следует заполнить следующим образом (id in (#VALUES#)) или аналогичным, при исполнении запроса #VALUES# заменится на список значений через запятую.

Поле SQL-выражение. Пишется либо #VALUE# либо пишется маска ввода, например для даты – to_date(‘#VALUE#’,’DD.MM.YYYY’). слово VALUE должно быть написано обязательно заглавными буквами.

Поле Настройка типа 2. В этом поле пишется функция, возвращающая корень дерева для параметра, формирующегося в виде дерева

Поле DISPLAY_VAL. Пишется имя пакета, точка, имя функции, возвращающей словесное имя. Например если необходимо вернуть имя объекта – utl_obj.get_name(:1)

:1 – т.е. функция подставит значение #VALUE# - данное поле имеет смысл заполнять только в случае типа параметра – rbs_lookup или rbs_lov.

2) Специальные параметры

Специальные параметры – имена этих параметров заранее запрограммированы, действия при задании таких параметров так же заранее известны, это сделано для облегчения настройки сложных параметров.

Параметр «Поиск по имени». Существует специальный параметр для представления - дерева – FIND («Поиск по имени»). Этот параметр осуществляет поиск по имени объекта. Данный параметр описан программно, поэтому его создание требует заполнения 4-х полей. Описывается как:

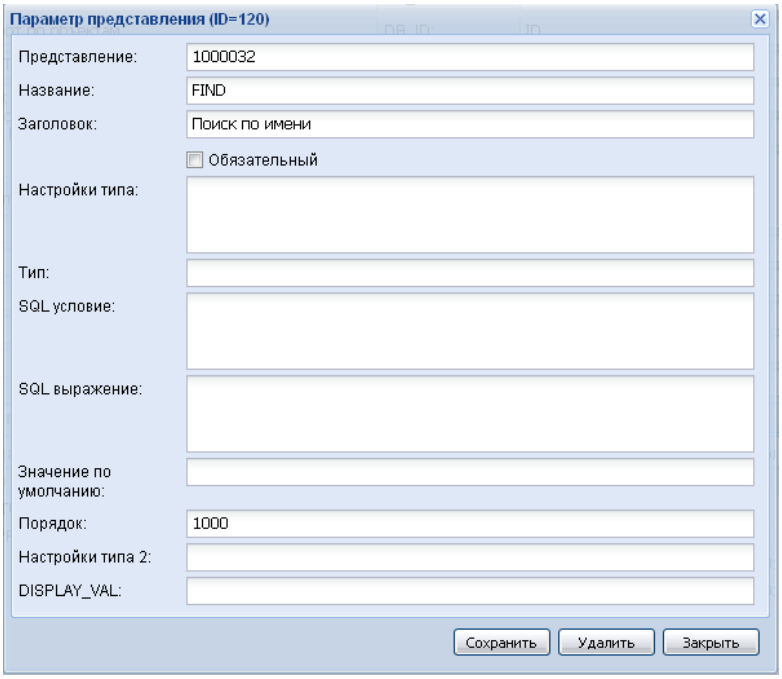


Рис. 21 Параметр «Поиск по имени»

Для редактирования параметра необходимо его выбрать на вкладке «Параметры представления» и дважды по нему кликнуть. Появится форма «Параметр представления», такая же, как и при создании.

Параметр «Корень дерева». Существует специальный параметр для представления – дерева ROOTID (Корень дерева), он необходим в том случае когда дерево велико и нужно работать с одной из его веток, если эта ветка постоянно одна и та же можно заполнить поле «Defaul root ID» (п.3.1.f) , если же ветки нужно периодически переключать используют этот параметр. Данный параметр описан программно, для его правильной работы нужно его описывать так:

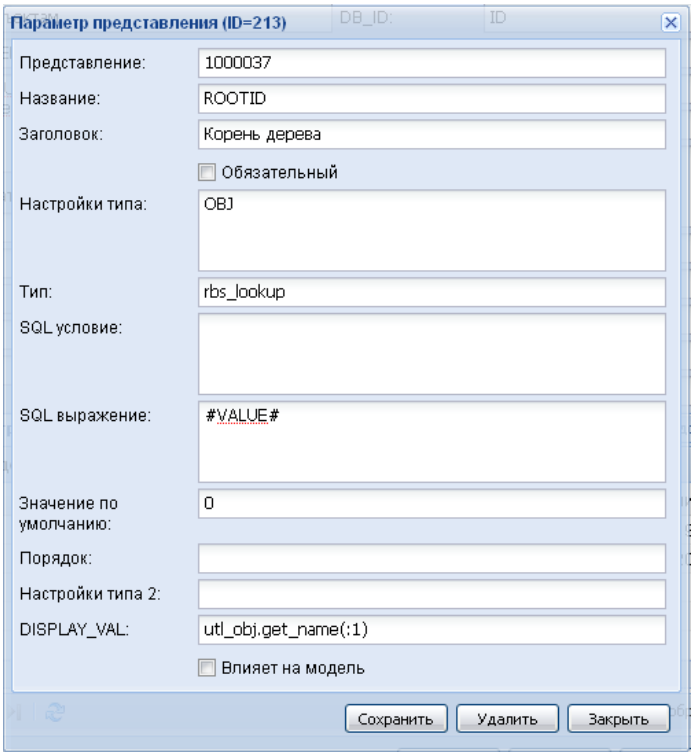


Рис. 22 Параметр «Корень дерева»

В поле «Настройка типа» необходимо указать представление-дерево из которого следует выбрать веерку.

У данного параметра ОБЯЗАТЕЛЬНО должно быть задано значение по умолчанию.

Поле DISPLAY_VAL необязательно к заполнению.

8.6 Создание кнопок

Для создания кнопки нужно открыть форму редактирования представления, открыть вкладку «панель инструментов». На панели инструментов вкладки, справа сверху есть кнопка – «Создать новый объект». Нажав на нее, откроется форма для создания/редактирования новой кнопки (рис 23).

Поле Сист. имя – системное имя кнопки при программном обращении к ней;

Поле Текст – тот текст, который будет отображен на странице представления;

Поле Иконка – путь к файлу с картинкой, которая будет отображена на странице представления;

Поле Подсказка – текст, который будет появляться при поднесении указателя мышки к данной кнопке;

Поле Функция js – функция, написанная на языке javascript, или же ее вызов. Для вызова, функция должна быть описана в специальной среде разработки.

Причем для каждой страницы существует индивидуальный файл с описанием необходимых для работы функций.

Если вы попытаетесь скопировать из другого представления какую либо кнопку, вам будет необходимо скопировать поле «функция js» - т.к. оно является ключевым в описании кнопки. Есть вариант задания функция – непосредственное написание тела процедуры на языке js. Для копирования таких кнопок достаточно скопировать содержимое поля «функции js». Другой вариант копирования возникает когда в данном поле есть только вызов функции, причем сама кнопка находится в другом представлении. В этом случае кнопка работать не будет, т.к. функция должна быть описана для конкретной страницы. Данный вариант описания возможен только с использованием специальной среды разработки.

Кнопки могут быть двух типов – кнопки, вызывающие формы, и кнопки, вызывающие окна. Окна – это представления, формы – это набор полей.

Поле Порядок – индекс, указывающий положение кнопки по счету, среди всех кнопок. Если коли

Если кнопка уже создана, то для ее редактирования достаточно дважды щелкнуть на ее имени в списке кнопок на той же вкладке «панель инструментов». Окно для редактирования такое же, как и для создания.

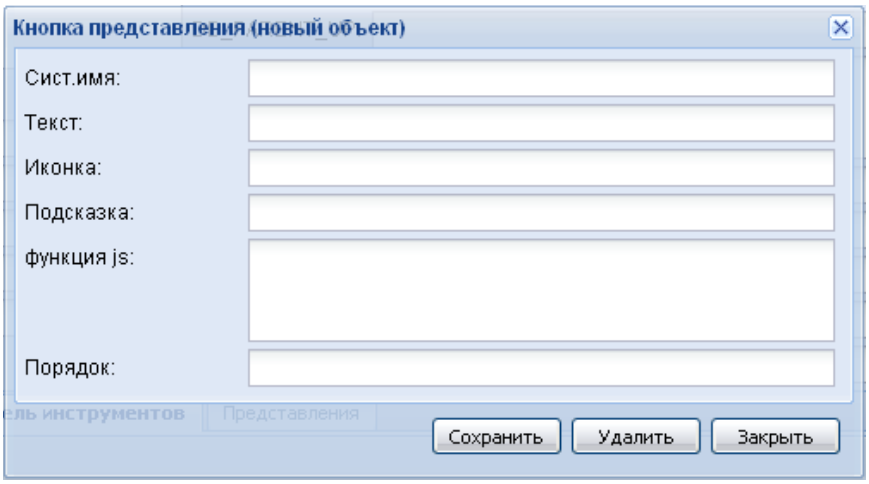


Рис. 23 Кнопка представления

8.7 Работа с данными представления

Данные представления можно редактировать, если это не запрещено при настройке.

При двойном клике на любом объекте, отображающем данные представления, возможно появление формы (рис. 24), была описана при

настройке, если при настройке указали построчное редактирование поле будет доступно для изменения (рис. 25).

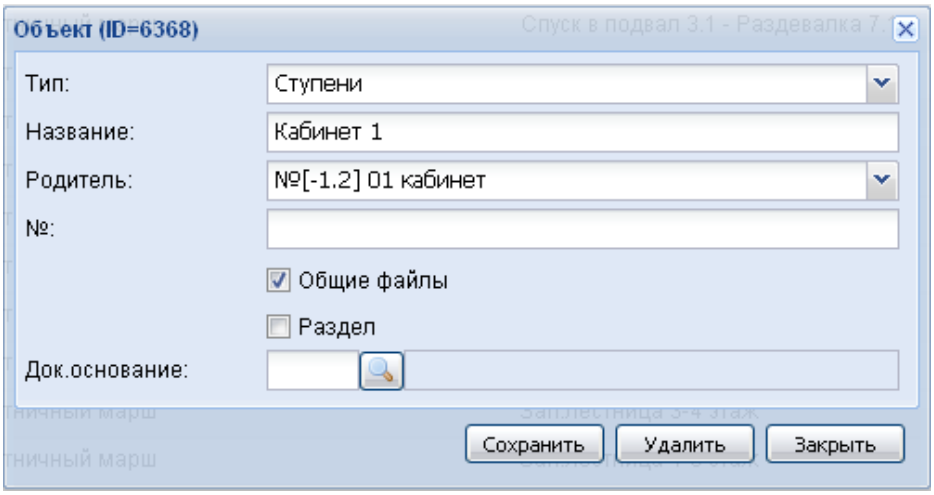


Рис. 24 Форма редактирования объекта

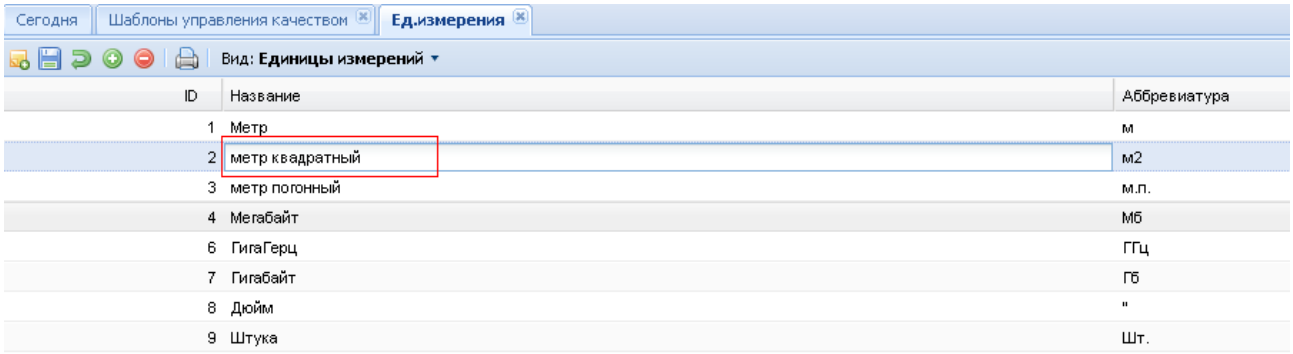


Рис. 25 Построчное редактирование

8.8 Дочерние представления

В представлениях есть возможность создавать взаимосвязанные представления. Причем есть родительские и дочерние.

Для создания взаимосвязанных представлений есть ряд условий:

- все поля должны быть одинаковы по описанию (но могут быть наполнены разными данными),
- тип отображения данных должен совпадать.

Что бы создать дочернее представление необходимо в форме редактирования представления в поле Родитель, нужно написать ID представления, которое станет родительским. При этом если зайти в форму

редактирования к родительскому представлению, во вкладке «Представления» отобразиться краткая информация о дочернем представлении.

8.9 Формирование запросов в представлении

В любом представлении происходит формирование запроса, т.к. для отображения данных их нужно получить из таблиц БД. Такое формирование происходит автоматически и оно различно, для разных типов отображения данных.

1) Формирование запросов для типа отображения данных Таблица

При формировании запроса с данным типом представления, все зависит от набора полей представления, его параметров, а также важно имя главной для данного представления таблицы (Схема 1).

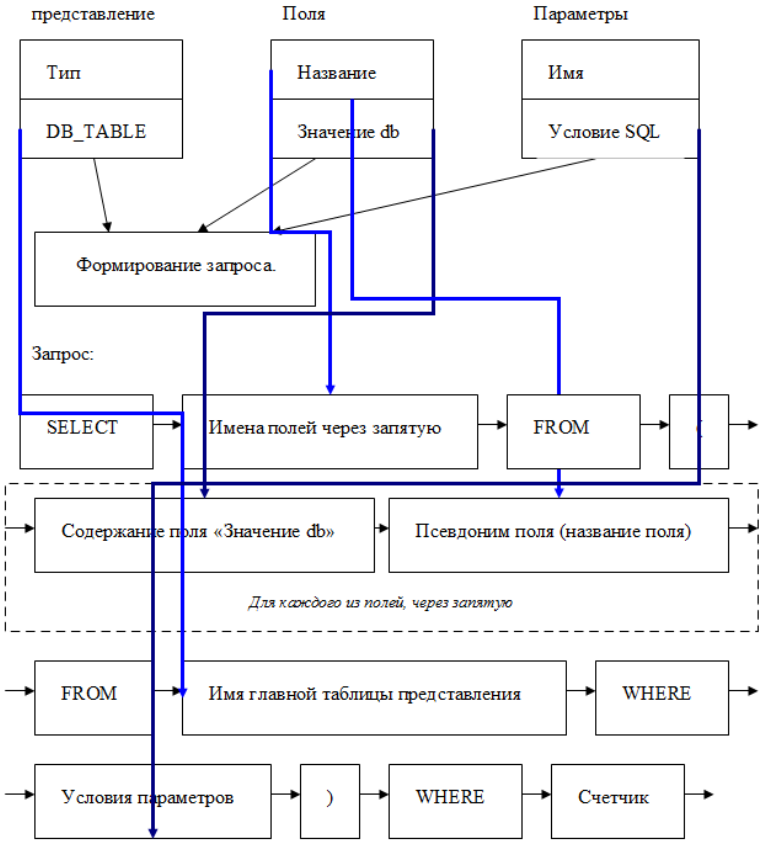


Схема 1 Формирование запроса для таблицы

На данном этапе разработки, выполнение сформированного запроса происходит при загрузке страницы, при обновлении представления, а также при переключении страниц представления. Выполнение запроса при переключении страниц представления не лучший вариант для работы с большим объемом данных, т.к. занимает много времени. Поэтому в дальнейшем следует изменить этот принцип работы.

2) Формирование запросов для типа отображения данных Дерево

Для данного типа отображения запрос пишется в поле «Запрос» в форме редактирования представления. Данный запрос пишется по определенному шаблону (Схема 2)

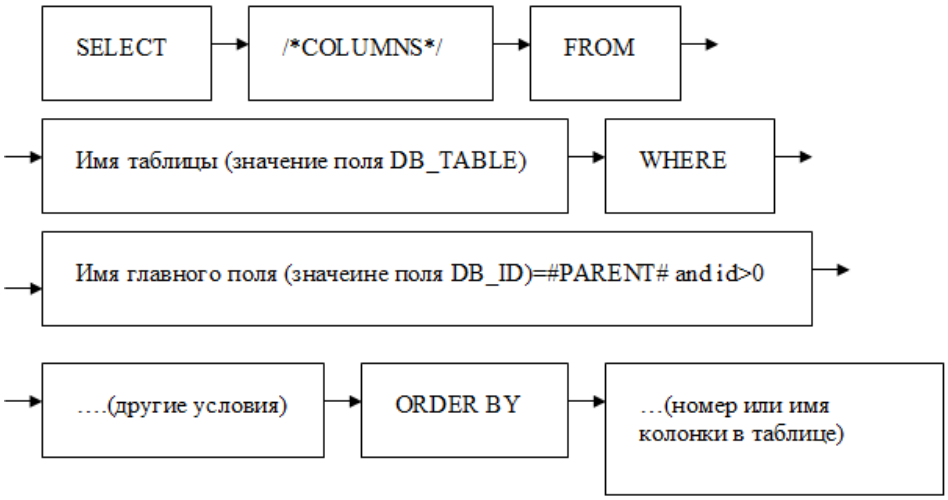


Схема 2 Структура запроса для представления – дерева

При формировании запроса программа заменяет слово /*COLUMNS*/ списком всех колонок, присутствующих в данном представлении, а слово #PARENT# значением ID родительского объекта.

Параметры при отображении данных в виде дерева не вставляются в запрос. Они могут быть использованы в функциях, вызываемых в колонках. Причем обращение к параметрам должно быть следующим: /*#<название параметры>_VAL#*/.

8.10 Chart для представлений

По структуре chart можно разделить на 2 группы:

- AreaChart, BarChart, RadarChart, ScatterChart – суть этих chart в том, что точка в них строится по координатам x,y (в линейной или полярной системе координат).
- PieChart, Cauge – эти chart по сути одномерные, то есть они визуализуруют соотношение нескольких однородных значение (например процентное соотношение).

По методу хранения данных Chart можно разделить на 3 группы:

- с непосредственной привязкой к представлению
- данные для chart получаются после применения функции javascript функции к данным представлениям

- данные для chart получаются непосредственно с сервера

1) Редактор используется преимущественно для создания chart первой группы, в этом случае команда выполнить будет строить chart с store замененным на store представления.

2) Отображение связанных chart Для представления добавляем метод createLinkedScreen(screenName, config), если задан config то будет применены к конфигу экрана или переданы как параметр в функцию конструктор, если экран реализован классом. Если экран и содержит useGridStore === true, то store для chart заменяется на store для grid. Если экран содержит функцию afterScreenRender(grid), то эта функция вызывается с первым параметром равным grid, для которого этот экран создается как chart.

В данной функции может быть выполнено построение store на основе store grid, добавление действий к grid и т.д.

3) Действие выполняемое при смене преставления в настоящий момент не определено. При смене представления выполняется смена store. Данное действие должно повесить на событие afterchangeview для представления, и будет изменяться в зависимости от конкретного графика.

4) Связанные chart задаются в параметре linkedScreens в параметрах представления, где нужно просто перечислить имена связанных экранов. Для создания экранов с этими chart можно использовать функцию createLinkedScreen(screenName, config)

5) По умолчанию если вызвать показать связанный экран через меню (находится в настройке представления), то связанный экран будет показан в окне.

9. Работа с рядами данных

Клиентская часть Редактора рядов (далее - РД) реализована в виде экрана SERIES_EDITOR.

На экране мы видим:

- Дерево с функциями (данные хранятся в таблице ANALIZ). Дерево Функций
- Две панели под деревом с функциями
  - левая показывает пустой график График
  - в правой панель с текстовым полем (Преобразователь) и пустое текстовый редактор (Ряд)

- Справа от дерева функций мы видим еще одно дерево (Исходные данные)
- Под Исходными данными видим еще одно пустое поле Информация по данным

Дерево Исходные данные является отображением дерева файловой системы начиная от <bmroot>/config/series/data и отображает все файла. Предполагается, что в данном дереве хранятся файла типа hdf5 с которыми работает библиотека pytables http://www.pytables.org/usersguide/datatypes.html . Каждый такой файл будет содержать данные одного или нескольких рядов (сейчас 1).

Таблицы внутри hdf5 файла имеют пути относительно абстрактного корня, в настоящий момент предполагается, что все файлы содержат 1 таблицу находящуюся по пути /table

Если выбрать вершину дерева являющуюся файлом нажать на кнопку Информация, то в текстовом поле Информация по данным будет отображена информация о структуре файла (какие группы, таблицы там есть, какие типы и поля таблиц)и данные таблицы /table.

9.1 Загрузка нового ряда

Загрузка ряда выполняется по следующему алгоритму:

1. Вводится имя файла
2. Выполняется загрузка csv файла
3. В режиме preview выполняется донастройка данных csv файла и полей h5 таблицы
4. При нажатии Закончить выполняется перенос сформированной h5 таблицы по указанному в имени пути
    - Для загрузки нажимает на значек + выбираем действие Загрузить csv
    - Вводим имя файла (становится активным поля для загрузки файла)
    - Загружаем файл
    - Если загрузка прошла без ошибок пользователь видит preview h5 таблицы и данные загружаемого файла(строки с ошибками подсвечиваются красным)
5. Пользователь может настроить имена колонок и типы, для этого нужно щелкнуть на имени колонки представления.

- Поднимается форма для редактирования колонки
- Можно изменить Имя, Тип данных, Функцию преобразования, Размер. Размер применяется только для строковых полей.
- При нажатии на кнопку Сохранить будет сохранен новый конфиг для колонки.
- Если колонок больше чем нужно, удалить их можно только правя config, который как и в предыдущем варианте доступен пользователю

6. Пользователь может изменять символ разделителя, символ экранирования и кодировку, а так же вносить изменения в данные csv файла в текстовом поле.

7. Когда настройка закончена нужно нажать кнопку Применить и новый конфиг будет применен для создания h5 таблицы по csv файлу.

8. Количество итераций донастройки не ограничено, после каждого нажатия кнопки Применить по данным конфига, и csv файла будет построена новая h5 таблица и результаты будут показаны пользователю.

9. Когда будет получена желаемая h5 таблица, нужно нажать кнопку Закончить для переноса созданной таблицы в нужное место.

- на нажимая кнопку закончить можно найти создаваемую таблицу в каталоге upload и перенести ее руками.

9.2 Загрузка в уже существующий ряд.

В уже существующий ряд можно добавить данные.

За загрузку в уже существующий ряд отвечает конфиг: params_for_apply

```
"params_for_apply": {
  "ifexists":  "add",
  "backup": "create",
  "compatibility": "force"
 }
```

На данный момент есть 3 параметра:

1. ifexists – допустимые значения “add”, “override”
   - Если задан “add” (по умолчанию) данные будут добавляться к уже существующим данным

- Если задан “override”, то файл с данными будет перезаписан (в том числе все настройки и все что хранится в файле ряда)

2. backup – допустимые значения “create”, “force_no_backup”

- Если задан “create” (по умолчанию) будет создан бекап изменяемого файла в каталоге _backup
- Если задан “force_no_backup” бекап создаваться не будет

3. compatibility – допустимые значения “force”, должен быть задан только если ifexists = “add”

- Если задан “force” – значит, что имена и типы в добавляемой таблице должны совпадать с именами и типа таблицы к которой выполняется добавление данных.

Процесс добавления происходит точно так же как и простая загрузка:

1. Загружаем данные

2. Приводим их к желаемому виду. Если должно быть выполнено добавление, то приводим таблицу к виду таблицы в которую добавляем (используем кнопку Применить)

3. Нажимаем кнопку Закончить для добавления данных в существующую таблицу.

Если все прошло хорошо, данные добавятся и окно загрузки закроется, в противном случае будет показано сообщение об ошибке и нужно будет исправить данные или конфиг.

9.3 Доступные для использования библиотеки Python.

На данный момент доступны следующие пакеты:

pandas - предназначен для работы со структурами данных

numpy - библиотека математических функций, поддержка многомерных матриц

scipy - библиотека для математических расчетов, возможность использования математических констант и специальных функций

scikit-learn - библиотека, включающая в себя различные алгоритмы для машинного обучения, в том числе позволяющая реализовать кластерный анализ, факторный анализ, построение регрессионных моделей, наивный Байесовский метод и пр.

statsmodels - модуль, который обеспечивает оценку различных статистических моделей, а также проведенит статистических тестов и исследования данных. Позволяет осуществлять построение регрессионных моделей и оценку их

параметров, дисперсионный анализ, анализ временных рядов, проводить тестирование ряда и пр, классы и функции с реализацией различных статистических моделей.

nltk - библиотека, предназначенная для обработки текстов на естественном языке.

9.4 Расчет статистики

Поддерживается 3 типа статистик:

1. Глобальные статистики. (G)

Данные статистики вычисляются по части данных, хранящихся в h5 таблице показателя ряда (Analiz, Показатель).

Для вычисления нужно задать диапазон: h[Header1]r1:h[Header2]r1.

Примеры:

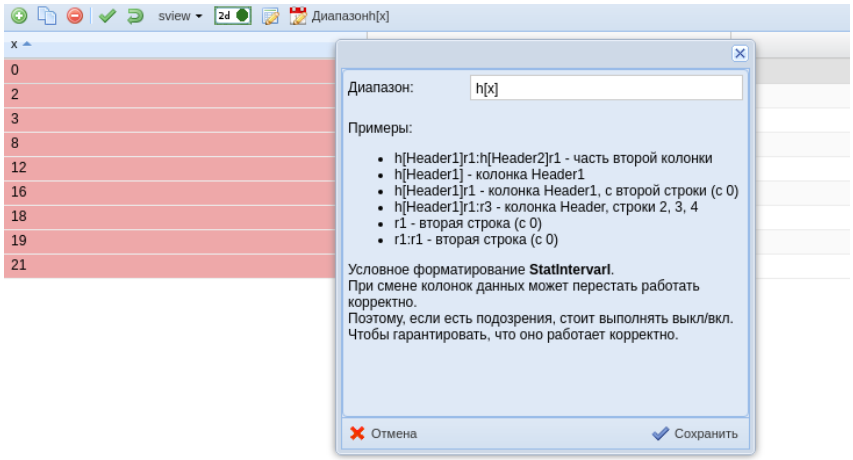
h[Header1]r1:h[Header2]r1 строка 2 с Header1 до Header2

h[Header1] Задать столбец

r1:r1 Строка 2 (отсчет с 0)

Глобальные статистики имеют ANALIZ_ID равный 0.

Параметры для вычисления глобальных статистик задаются по кнопке Диапазон на вкладке Данные.



Доступна подсветка выбранного интервала с помощью условного форматирования StatInterval.

2. Локальные статистики. (L)

Такие статистики пишутся специально под h5 таблицу с данными ряда. (Analiz, Показатель)

Эти статистики имеют ANALIZ_ID равный id соответствующего ряда (Analiz).

3. Статистики папки (F)

Такие статистики аналогичны локальным статистикам (см. п2), но их ANALIZ_ID равен ID папки, в которой находится ряд. (Analiz).

Для вычисления таких статистик нужен ANALIZ_ID ряда, для которого должно выполниться вычисление. ANALIZ_ID присылается при запросе пересчета с браузера.

9.5 Создание новой статистики

Для задания локальной статистики по ряду нужно:

1. Перейти на вкладку Анализ показателей статистики
2. Создать новую статистики, задать Название, Описание.
3. Задать Функцию расчета, например:

```
def compute_stat(hdf5table, compute_stat_api):
    return len(hdf5table.get_data())
```

Функция должна называться compute_stat, на вход она принимает hdf5table объект и compute_stat_api.

Для расчета, как правило, достаточно:

- hdf5table.get_data – получить список словарей с данными
- hdf5table.table – получить объект tables.Table, документация для него есть на pytables.org

Создание статистики для папки выполняется аналогично, только нужно чтобы форма Analiz была открыта на нужной папке. Либо можно вручную изменить ANALIZ_ID на id требуемой папки.

Создание глобальной статистики выполняется аналогично локальной. За исключением того, что:

1. Нужно указать в качестве ANALIZ_ID 0.
2. Функция compute_stat принимает на вход последовательность данных, а не объект hdf5table.

Например:

```
def compute_stat(data_series, compute_stat_api):
    """ Compute average value for data series
    """
    if not hasattr(data_series, '__iter__'):
        raise Exception("Not serie at input!")
    if len(data_series) == 0:
        print ("average = %s" % 0)
        return 0
    serie_sum = float(sum(data_series))/float(len(data_series))
    print ("average = %s" % serie_sum)
    return serie_sum
```

Второй аргумент compute_stat_api можно использовать для получения, h5 таблицы методом compute_stat_api.get_hdf5table. В настоящее время поддерживается, только один метод, в дальнейшем при необходимости compute_stat_api может быть расширен.

Например так может быть реализована глобальная статистика расчета длины данных:

```
def compute_stat(data_series, compute_stat_api):
  """ Compute length of data series.
  """
  return result
  result = len(compute_stat_api.get_hdf5table().get_data())
```

9.6 Изменение колонок таблицы

Для таблиц в которых хранятся ряды данных есть возможность изменения имени и конфига колонки, а так же добавление и удаление новых колонок

При этом нужно учитывать ряд ограничений:

Поскольку ряды хранятся в hdf5 таблицах, в которых задается жесткая структура данных, изменение колонок таблицы подразумевает под собой создание копии таблицы 2. Если выполняется преобразование типа из float или во float возможны искажения данных, из-за особенностей хранения чисел с плавающей точкой в компьютере, например float: 8.8 может стать строкой

8.800000092 3. В pytables None значения не поддерживаются, поэтому они эмулируются:

- для строковых полей это строка “None”
- для числовых полей это -999999999
- для float это диапазон [-999999999 - 10, -999999999 + 10]
- для остальных типов None не определено

Возможность использовать только dflt значения вместо None в настоящее время не реализована.

Поскольку копирование может быть продолжительным, за одно преобразование таблицы можно выполнить сразу несколько изменений.

Изменения задаются конфигурацией, называемой action, каждый action это словарь.

9.7 Конфигурация action

action - должно быть задано что-то из CREATE, CHANGE, DELETE (обязательно)

fieldName - имя изменяемого поля, либо имя создаваемого поля (обязательно)

newFieldName - новое имя поля, задается только если выполняется переименование

fieldConfig - строка описывающая колонку таблицы задается для создания, и изменения, если меняется конфиг колонки

В интерфейсе настройка выполняется в форме HDF5_TABLE, которая поднимается например по dblclick в информационном фонде, на форме доступна кнопка Настроить колонки, по нажатию на которую поднимается форма HDF5_TABLE_CONFIG, в которой нужно задать конфиг действий, и нажать кнопку Применить

Перед выполнением изменений таблицы можно выполнить сохранение файла, на случай если результат преобразования будет не удовлетворительным.

9.8 Описание структуры конфига

Пример конфига:

```
{
    "table": {
```

```
    "filename": "time.h5",
    "tablepath": "/table",
    "table_config": {
        "fields": {
            "Date": "Int64Col(pos = 0)",
            "Number": "Int32Col(pos = 1)"
            }
        },
        "description": "Описание ряда"
    },
    "data": {
        "type": "csv",
        "start_skip": 2,
        "fields": ["Date", "Number"],
        "delimeter": ";",
        "quotechar": "|",
        "getDate": "getIntFromRusMonthYear",
        "getNumber": "getInt"
        }
}
```

Конфиг должен содержать 2 поля table, data.

1. table – определяет куда загружать данные
    - filename имя файла
    - tablepath путь в файле, сейчас можно ставить /table
    - table_config нужно заполнять, если файл еще не создан, если создан то поле не обязательно
        - fields – объект в котором определены поля таблицы в формате <name>: <функция создания>

- http://www.pytables.org/usersguide/libref/declarative_classes.html можно сделать поиск <some>Col, чтобы найти все поддерживаемые колонки.

- Как правило это будет StringCol(<size>), Int32, Int64, Float64, Time, … (для хранения дат сейчас используем Int64), при создании колонки можно указывать позицию.

◦ description – описание для ряда, будет отображаться в дереве таболиц (можно задать потом с использованием формы)

Для преобразования доступны функции getInt, getFloat, getIntFromRusMonthYear, getIntFromYear Можно загружать данные по месяцам и по годам с сайта http://cbsd.gks.ru/ , там есть кнопка сохранить, где можно выбрать csv.

2. data – определяет как будут считываться данные из csv файла,

- type указываем csv
- start_skip сколько пропустить сначала
- fields – как будут называться поля, нужно чтобы названия соответствовали полям table
- delimeter – разделитель в csv файле как правило ;
- quotechar – символ экранирования как правило |
- get<fieldName> – имя функции которые используются для преобразования строки csv файла в то, что будет хранится в таблице, в примере указаны 2 сейчас поддерживаемые функции

После загрузки файла, если его еще не было в дереве он должен появится.

9.10 Анализ рядов

Анализ выполняется на основе дерева, которое задается в таблице анализ, доступно:

- из формы DOC_GOODS_ANALIZ на вкладке модель данных,
- на вкладке Расчет для формы Анализ.

В таблице доступны два виде вершин (в дереве они никак не отличаются), вершины функции, вершины ряды.

Вершина ряд определяется тем, что первая строка функции расчета “def compute(self):”

Внимание! задание названия переменной для Вершины ряда обязательно.

На формирование кода и вычисление оказывают влияние следующие флаги устанавливаемые на форме:

- Отображать на графике – если флаг выставлен, то результат вычисления ряда, попадет в результирующие данные по вычислению и может быть отображен на графике
- Обязательно для вычисления – если флаг стоит, то даже если ряд, не используется в при вычислении основных данных, он будет вычислен. Данный функционал позволяет выполнять проверки и дополнительные действия при вычислении, которые не должны отображаться в основных результатах.

Для Вершин рядов можно задавать вспомогательные методы для расчета и использовать их как методы self.

После вычисления выполняется определение максимальных длин строковых колонок, так как это нужно для построения конфига таболицы хранения данных.

Чтобы немного ускорить данное вычисление можно вернуть результат вычисления в виде словаря формата:

```
{
  "result": result,
  "max_lengths": [-1, 0, 10, 100]
}
```

Где result это список или кортеж (list, tuple) с результатом расчета, а max_lengths список содержащий максимальные длины. Эти длины будут задавать длины строковых колонок генерируемой таблицы.

По созданным функциям будет построен модуль, который можно посмотреть нажав на Показать код.

9.11 Генерация кода

Код генерируется по шаблону:

1. Сначала размещается шаблон импорта и инициализации используемых для расчета переменных
2. Добавляется код для всех вершин функций
3. Объявляются классы для вычисления рядов, где в функцию compute подставляется указанный пользователем код
4. После создания класса создается экземпляр этого класса с именем указанным в поле Имя переменной формы

5. Создается экземпляр класс SeriesTableCombiner которому передаются все экземпляры вершин рядов
6. Выполняется вычисление рядов

Если все методы заполнены правильно, и при просмотре кода не было найдено недочетов, можно нажать Расчитать, будет выполнен код и будет показан результат выполнения (Без ошибок/С ошибками), информацию по выполнению расчета можно посмотреть нажав на кнопку Лог.

Результат будет сохранен в h5 файле имя которого определяется на основе номера отчета и типа отчета, в соответствующем представление можно посмотреть получившуюся в результате вычисления таблицу.

Для показателя результат расчета сохраняется в файл имя которого можно настроить в поле Конфиг определяющий путь таблицы, на вкладке Импорт. По умолчанию будет characteristics/<VARIABLE_NAME>/table.h5. Результирующую таблицу можно посмотреть на вкладке Данные.

9.12 Соединение рядов

Когда при расчете используется несколько рядов, то чтобы получить результат расчета нужно выполнить соединение данных нескольких рядов в 1.

Соединение (join) выполняется на основе информации о ключевых значениях. У каждого ряда есть ключевые значения колонки и не ключевые. Ключом результирующего ряда является ключ максимальной длины.

По умолчанию(если пользователь не указал явно) считается что: 1. если у ряда 1, 2 колонки то 1-ая колонка является ключевой 2. если у ряда больше 2 колонок, то первые 2 являются ключевыми.

Задать ключевые колонки можно задав:

```
def get_key(self):
  return set((0, 2))
```

Где 0, 2 колонки которые считаем ключевыми.

Если у всех рядов ключи одинаковые, то слияние проводится одинаково, берутся все возможные комбинации имеющихся ключевых значений и для них определяется значение результирующего ряда путем выбирания значений соответствующего ключа для каждого из рядов.

Если же длины ключей разные, то есть два варианта:

1. Если длина ключа для ряда меньше длины ключа результирующего ряда, то недостающие ключевые значения выставляются в None и таким образом формируется ключевое значение.
2. Если длина ключа для ряда меньше длины ключа:

   Например:

   a, b

   При ключе a, b, c, d.

   То считаем что значения ряда в точке a1, b1, c1, d1 равно значению ряда в точке a1, b1, то есть для всех недостающих частей ключа значение константа.

Вариант 2 является вариантом по умолчанию так как он более логичен.

Для изменения значения можно вставить в какую-нибудь из серий:

```
def compute(self):
    return [1]
# изменение compute_config['key_join_type']
compute_config['key_join_type'] = 'simple'
```

Либо создать еще объект анализ с функцией где изменить значение.

9.13 Расчет для показателя

Расчет для показателя выполняется на вкладке Расчет формы Анализ.

Формирование и выполнение расчета выполняется по тому же алгоритму, по которому выполняется расчет для полосы отчета.

Разница заключается в выборке подрядов включаемых в расчет, для этого выбираются все дочерние вершины для рассчитываемой (В полосе отчета выбираются все вершины которые ссылаются на полосу отчета и дочерние).

Результат расчета сохраняется в файл данных полосы отчета полосы отчета.

Для показателя местоположение файла с данными определяется свойством PY_TABLE_PATH_CONFIG.

Свойство PY_TABLE_PATH_CONFIG (В форме Анализ Конфиг определяющий путь таблицы) может задаваться следующим образом:

1. Ничего не задано.

   В этом случае данные хранятся в файле characteristics/<VARABLE_NAME>/table.h5.

Все остальные данные в characteristics/<VARIABLE_NAME>.

2. Путь до hdf5 файла (например /series/table.h5)

   В этом случае файл с данными хранится по указанному пути.

   Все остальные данные в каталоге characteristics/<variable_name>

3. Json конфиг.

   Поддерживаемые параметры: relative_path, h5_file_name, py_table_path, folder_path

- folder_path – путь где хранятся все данные по показателю (по умолчанию characteristics/<variable_name>). Таблица данных может хранится отдельно.
- <relative_path>/<h5_file_name> – путь до файла с данными (по умолчанию characteristics/<variable_name>/table.h5)
- py_table_path путь до таблицы в hdf5 файле. (по умолчанию /table)

9.14 Доступ к показателям по имени

Для доступа к данным и таблицы показателя по имени или номеру можно использовать функции:

data_utils.get_analysis_data_by_name

data_utils.get_analysis_data_by_id

data_utils.get_analysis_table_by_name

data_utils.get_analysis_table_by_id

9.15 Доступ к базе данных

Словарь compute_config предоставляет свойство connection (экземпляр DatabaseConnection), для работы с базой данных.

```
#пример обращения к базе данных
connection = compute_config.get("connection")
req = "select * from analiz where rownum < 3"
res = connection.exec_get_list(req) #<-- DatabaseConnection instance
print "res = ", res
```

9.16 Доступ к данным документа, листа и анализа

Для доступа к данным листа, данным документа, данным анализа доступны следующие функции:

```
# получить данные листа отчета0
```

```
rep_list_props = self.get_obj_properties()

# получить данные отчета
doc_props = self.get_doc_properties()

# получить данные текущей вершины анализ
analysis_props = self.get_analiz_properties()

#так можно вывести данные в логе, для удобного просмотра
import pprint
print 'doc_goods_properties = ', pprint.pformat(doc_goods_properties)
```

Стоит учитывать, что get_obj_properties, get_doc_properties сработают только для вершин ANALIZ, с заданными OBJ_ID, OBJ_TYPE_ID.

Как правило такая вершина является корнем Дерева вычислений полосы отчета. Для дочерних вершин это не обязательно верно, для таких вершин можно вызывать указанные свойства корневой вершины.

Например:

```
# получить данные листа отчета
rep_list_props = ROOT_VARIABLE_NAME.get_obj_properties()
```

В коде compute вершины.

9.17 Мониторинг результатов вычисления

При выполнении расчета возможно установить значение флага и значение сенсоров. Установленные значения могут использоваться в представлениях или других частях bm для привлечения внимания к определенным отчетам.

Для сигнализации при выполнении расчета добавлен объект notifier (уведомитель), ползователю доступны 3 метода:

- set_notify_value (value)

Метод устанавливает свойство NOTIFY_VALUE DOC_GOODS для которой выполняется отчет в значение VALUE, которое потом может быть использовано как метка, что на эту товарную партию нужно обратить внимание

Получить значение в представлении можно например так:

select utl_attr.get_value(3669, 1996, utl_ot.get_attr_id(3669, 'NOTIFY_VALUE')) from dual;

3669, 1996 соответственно номер типа и номер товарной партии могут быть заменены на параметры

- save_sensor (value, value_type, text, dt_sensor)

Добавляет запись в sensor_data, в которой OBJ_TYPE равно типу товарной партии OBJ_ID номер товарной партии VALUE, VALUE_TYPE значение и тип значения сенсора TEXT текст DT_SENSOR можно записать дату полученную при вычислении DT дата выполнения вычисления

Если использовать эту функцию, то при каждом вычислении в sensor_data будут записываться новые значения

- save_or_update_sensor (value, value_type, text, dt_sensor, keys)

Метод пытается найти запись с максимальным id за максимальную дату, у которого OBJ_TYPE, OBJ_ID тип и номер товарной партии DT_SENSOR, VALUE, VALUE_TYPE равны переданным значениям, проверяются только те свойства что есть указаны в keys, если keys не указано, считаем что в keys только DT_SENSOR Если запись не найдена, метод будет работать как save_sensor

Использование этого метода позволяет при повторных вычислениях выполнять обновление установленных сенсоров

Выбор данных из таблицы sensor_data можно выполнять стандартными способами

9.18 История изменений

Таблицы поддерживают ведения лога изменений.

Для таблицы поддерживается три уровня ведения логов:

1. Без логов, как следует из названия логи при этом не ведутся (по умолчанию)
2. Упрощенный, в качестве лога записывается только информация о времени изменения, пользователе и действии SA, SU, SD (добавление, обновление и удаление) и количество строк которые затронули изменения.
3. Полный, записывается пользователь, время действия, действие A, U, D (добавление, обновление, удаление), номер строки для которой было выполнено действие и если это обновление или удаление сохраняется исходная строка таблицы.

Действие добавить, добавляет строки в конец таблицы, поэтому в логе для данного действия сохраняется только количество добавленных строк.

Включить ведение логов можно:

- непосредственно в представлении, для этого нужно:

- перейти на вкладку Таблица в параметрах,
- выбрать нужный режим логирования
- нажать Применить параметры к таблице.

  Данный параметр не влияет на отображение данных представления поэтому для него сделана отдельная кнопка.

- при создании таблицы через загрузку csv файла в основных параметрах выбрать режим логирования.

Для просмотра истории в представление добавлена кнопка История.

9.19 Периодический пересчет данных ряда

Для рядов можно настроить автоматический пересчет, чтобы это сделать нужно:.

1. Поставить галочку в поле Пересчет активен в форме настройки отчета (В текущей реализации на вкладке Служебная)
2. Поставить галочку в поле Пересчет активен в форме настройки полосы отчета (В текущей реализации на вкладке Служебная)
3. Настроить конфиг пересчета, для полосы отчета.

Настройка конфига периодического пересчета ряда выполняется в форме, которая поднимается по кнопке Пересчет на вкладке Модель данных формы настройки ряда (форме настройки полосы отчета).

В форме нужно заполнить поле Параметры расчета, в виде json. Например:

```
{
   "period": "*/30 * 3 * *",
      "delta": 300,
}
```

Единственный поддерживаемый на данный момент параметр это “period”, его значение должно быть строкой, которая задает параметры пересчета аналогично заданию в утилите cron: https://ru.wikipedia.org/wiki/Cron

То есть первое значение это минуты, второе часы, третье день месяца, четвертое месяц, пятое день недели.

Все значения начинаются с 0.

Символ *, означает любое значение, строка */n, означает, что соответствующее значение должно быть кратно n.

В приведенном выше примере пересчет будет вычисляться каждые полчаса каждого 3-его числа месяца.

Просмотреть результаты периодических расчетов можно по кнопке Расчеты логи На вкладке Витрина данных -> Модель данных формы настройки расчета ряда. Либо двойным нажатием на поле Время расчета вкладки Настройка отчета формы настройки отчета.

Параметр delta используется для задания количества секунд, которое должно пройти с момента предыдущего расчета, прежде чем можно будет запустить новый расчет.

При выполнении расчета каждые 30 секунд в базу данных заносится информации о текущем времени, и в течении следующих 60 секунд считается что выполняется расчет и данные расчет не может быть запущен автоматически. Это позволяет запускать сервис выполнения периодических расчетов в несколько процессов или на нескольких серверах выполняя в каждый момент времени не боле одного расчета.

Интервалы времени задающие ограничения могут быть изменены в настройках.

Ограничением является то, что время на серверах должно быть синхронизировано.

10. Завершение работы с Программой

Завершение Программы пользователь может осуществить через закрытие соответствующей вкладки браузера или нажав на кнопку “Выход”

11. Возникновение вопросов или внештатных ситуаций

При возникновении вопросов или внештатных ситуаций в процессе эксплуатации Программы, необходимо обращаться к техническому специалисту:
bm_support@t48.ru.

+7 (495) 915-02-44 доб. 760